Social Network Analysis and Mining (2022) 12:72
https://doi.org/10.1007/s13278-022-00889-6

ORIGINAL ARTICLE

Social data provenance framework based on zero-information loss graph database

Asma Rani[1,2] · Navneet Goyal[2] · Shashi K. Gadia[3]

Received: 3 September 2021 / Revised: 11 March 2022 / Accepted: 5 May 2022 / Published online: 3 July 2022

Abstract
Social media has become a common platform for global communication across the world due to its rapid dissemination of information among a large audience. Its popularity has raised a crucial challenge to capture the social data provenance of a piece of information published on social media. Social data provenance describes the source and deriving process of a digital content, and when it is updated since its existence? It aids in determining reliability, authenticity, and trustworthiness of a piece of information and explaining how, when, and by whom this information is published. In this paper, we propose a social data provenance (SDP) framework based on zero-information loss graph database (ZILGDB). The proposed framework supports historical data queries, and querying through time using updates management in ZILGDB. It has the capability to capture provenance for a query set including select, aggregate, and data update queries with insert, delete, and update operations. It also provides a detailed provenance analysis through visualization and with efficient multi-depth provenance querying support, to determine both direct and indirect sources of a digital content. We conduct a real-life use case study to evaluate the usefulness of proposed framework in terrorist attack investigation. We evaluate the performance of proposed framework in terms of average execution time for various provenance queries, and provenance capturing overhead for a query set.

Keywords Social media · Social data provenance · Graph database · ZILGDB · Provenance querying

1 Introduction

In present digitized world, everyday a huge amount of digital content is generated by various social media platforms mainly through mobile devices and shared with other users (Kaplan and Haenlein 2010; Corsar et al. 2016; Feng et al. 2018). In this way, social media has been playing a major role in information sharing at a large scale due to its easy access, low cost, and fast dissemination of information. However, social media has a positive impact in expanding society's access to information as well as it quickly amplifies the influence of digital content (Soni et al. 2019). However, no one cares about the truthiness, quality, and credibility of that content (Soto et al. 2018; Yang et al. 2019). We no longer know where this content comes from and whether it is correct or not? Moreover, the "publish first ask later" strategy of social media users is also a cause behind the dissemination of false information. In this context, identification of the source of a digital content and its history has become a crucial challenge in the field of social media analytics. Hence, to rebuild the trust, there is a need to restore the concept of data provenance (Buneman and Tan 2019; Cheney et al. 2009; Glavic and Miller 2011; Bearman and Lytle 1985; Herschel et al. 2017; Buneman et al. 2000; Buneman and Davidson 2010; Simmhan et al. 2005; Yuan et al. 2018) in social media. Social data provenance is the only way through which social media can regain some semblance of trust from their users (Corsar et al. 2016; Feng et al. 2018; Markovic et al. 2013; Riveni et al. 2017). In this paper, our main motivation is to explore the need of provenance data associated with digital content published on social media, and to design an efficient *social data provenance (SDP)*

✉ Asma Rani
asma.sags@gmail.com

Navneet Goyal
goel@pilani.bits-pilani.ac.in

Shashi K. Gadia
gadia@iastate.edu

[1] Department of CSE, DBRAIT, Port Blair, A & N Islands, India

[2] CSIS Department, BITS Pilani, Pilani, Pilani, India

[3] Department of CS, Iowa State University, Ames, IA, USA

framework based on *zero-information loss graph database (ZILGDB)*. Additionally, our motivation behind tracing the origins of digital content is to provide true sources to all the social media users of what they see, read and hear on social media.

1.1 Need of social data provenance

Today's *Social Networking Sites (SNS)* such as Twitter, Facebook, and Instagram are based on the tenets of mobilization and de-contextualization of information, where each user stands at the edge of a river of information to pick an independent data for either repost or sharing with other users without including ownership and description of the content. In such scenario, it is very crucial to perceive what is the origin and deriving process of an information, and when this information is updated since its existence? Social data provenance is not only associated with a social media data's history in time, but also with the relationships of this data with other entities which enabled its creation. The concept of social data provenance (Markovic et al. 2013; Riveni et al. 2017) can be used to answer the following questions viz., *What* is the source of this socially published data?, *Why* this data is generated?, *How* this data is derived?, *Who* has published this data?, *What* is the history of this data? etc. The whole idea of social data provenance is about quality, credibility, and trust of digital contents published on social media (Soto et al. 2018; Yang et al. 2019). However, social data provenance is an efficient tool for combating misinformation and to create a more trustworthy environment on social media, but zero or very minimal provenance information is provided by such social networking sites that are not sufficient to restore the trust and to improve the information literacy (Feng et al. 2018). Consequently, a provenance capturing framework for social media can be beneficial in building a more transparent environment for users by answering the following questions; *when, where and by whom a piece of information is published? How it is transformed? Who is the owner of a digital content? and how it has been spread across the network?*

1.2 Challenges

In recent years, social data provenance has gained a lot of attentions, as it serves different purposes such as audit trail, data discovery, update propagation, incremental maintenance, rumour identification (Duong et al. 2017), network extraction (Afra and Alhajj 2021) and justification of a query result, etc. In social media analytics, the credibility of an analysis is largely relying on the quality and trustworthiness of a digital content which is assured by social data provenance (Soto et al. 2018; Yang et al. 2019). In this way, social data provenance plays a major role in determining the amount of trust, authenticity, and quality of a digital content. However, extraction of such provenance information from a huge volume of unstructured social data is an extremely difficult task, because of their heterogeneous formats. In this paper, we identified following issues to address the key challenges in capturing, storing and querying provenance data associated with digital contents published on social media:

- Currently, no any social media platform is providing a meaningful context about the published information to ensure the credibility of that information.
- Most of the existing provenance solutions are not scalable to track provenance metadata for social media efficiently. They are suitable to capture workflow provenance at coarse-grained level only.
- Although, a few frameworks capture fine-grained provenance, yet they do not provide provenance support for all types of queries such as select, aggregate, historical and data update queries.
- There is no such provenance solution which has the potential to trace out the provenance path and propagation history of a digital content published on social media.
- Although, social data is produced in different formats, viz. text, multimedia file, image etc. However, there is no common format of such metadata is provided to understand the provenance information associated with digital content published on social media.
- As digital content published on social media is constantly changing over time, yet no any existing framework is capable to capture provenance for historical queries, which is an essential requirement of social data provenance.
- There is no common application programming interface (API) and widely accepted generalized architectural solution to access and manage provenance data associated with digital content published on different social media platform.

1.3 Relational versus graph database

Relational databases have been the mainstay of the data community for decades (Silberschatz et al. 1996), but these databases are not scalable for handling large size unstructured social media data. Graph databases have been proposed as an alternative to SQL databases to handle the challenges posed by social media, as these databases are very flexible by nature (i.e., dynamic schema), extremely scalable, fast in querying and focuses on relationships among relevant data (Angles and Gutierrez 2018; Robinson et al. 2015). In contrary to relational database, graph database is schema free and flexible, as every node in a graph may have different number of attributes and may be associated with different number of nodes via explicit relationships. Hence, graph databases are well-suited to store, process, and query graph-oriented data such as social data generated from social media. Therefore, in this paper, we attempt to design and develop a social data provenance framework for graph database using the concept of zero-information loss database.

1.4 Research contributions

Conventional databases are ill-suited for update management as they capture only present state of data without preserving any data updates. The proposed framework is developed around a novel concept of zero information loss database (ZILD) (Bhargava and Gadia 1993; Rani et al. 2015, 2016). By a ZILD, we mean that no data value, no user, and no query and its result (seen at the time query was issued) is ever lost. It is a special kind of temporal database that maintains temporal data as a history of all the updates along with the complete information of operational activities performed in that database. Therefore, it is well-suited for designing a provenance framework specially to capture provenance for insert, delete, update, and historical queries. In response to the above challenges, we design and develop a social data provenance (SDP) framework based on zero-information loss graph database (ZILGDB) (Rani et al. 2021). The proposed framework is capable to answer the following questions, viz. What type of provenance data should be captured from social media? At which extent it will be useful? How to capture and store this provenance data? Where to store this data? How to visualize/analyze this data? etc. In summary, the main contributions of this paper are as follows:

1. SDP framework is a novel provenance solution that captures both fine-grained and coarse-grained provenance associated with the digital contents published on social media.
2. Fine-grained provenance is captured in a form of provenance graph G_p (V_p, E_p) where 'V_p' is the set of nodes and 'E_p' is the set of edges, while coarse-grained provenance is captured in form of query statements with their execution time.
3. ZILGDB is a special kind of temporal database that provides data version support to maintain the history of all updates in form of provenance data along with the provenance of all insert and delete operations.
4. The proposed framework is capable in capturing provenance for all queries including select, aggregate, and historical or past queries, i.e., queries which were executed in the past. The captured provenance is stored in a provenance graph database (PGDB) for further analysis.
5. The proposed framework also supports to capture provenance for all insert, delete, and update queries.
6. To support provenance querying for historical data, we designed following four User-Defined cypher query constructs (UDCs) viz., "instance", "all", "valid_on now", and "valid_on date".
7. The framework also supports efficient multi-depth querying of provenance data to explore various direct and indirect sources of a piece of information.
8. Stored provenance can be analyzed through visualization for various purposes such as justifying the result tuple of a query, querying historical data, audit-trail etc.
9. Existing approaches are dedicated to a particular social media platform with limited query support. On contrary, our framework provides a generalized provenance solution for various social media platforms such as Twitter, Facebook, and Instagram.
10. As a use case, we use the framework to investigate terrorist attack by identifying suspicious persons and their linked communities using Twitter data.

2 Related work

Social media analytics is an emerging field that extract meaningful insight from an extensively large volume of social data. Social data provenance is often referred as the information about who created or changed content, what and how it was changed, regardless of their geographic location or access technology. The importance of social data provenance in social media analytics with several key challenges such as measuring quality and truthiness of social data, provenance capturing, provenance storage, and querying provenance are presented in Feng et al. (2018), Markovic et al. (2013), Riveni et al. (2017), Baeth and Aktas (2017), Tas et al. (2016), Cheng et al. (2012) and Wang et al. (2015). An open-source tool, i.e., Social Feed Manager (SFM) (Kerchner et al. 2016) is developed to capture metadata associated with a tweet. A RAMP model (Park et al. 2011) is proposed for Generalized Map and Reduce Workflows (GMRWs) using a wrapper-based approach for provenance capturing and tracing. Further, HadoopProv model (Akoush et al. 2013) is introduced for provenance tracing in Map Reduce workflows, where it traces the provenance in Map and Reduce phases separately and construction of provenance graph is deferred at query stage to minimize temporal overhead. In order to preserve temporal information, a data model for time-varying social media data (Cattuto et al. 2013; Durand et al. 2017) is proposed using Neo4j (NoSQL graph database). Cypher query language (CQL) is used to analyze a huge volume of twitter dataset (Soni et al. 2019; Boselli et al. 2017; Zhang et al. 2017). These research work explores the suitability of Neo4j graph database for both efficient storage, and fast query processing of social media data (Boselli et al. 2017; Zhang et al. 2017; Angles and Gutierrez 2008; Fernandes and Bernardino 2018; Sharma 2015; Allen et al. 2019). A web-based tool (Gundecha et al. 2013b; Ranganath et al. 2013) is developed to capture predefined provenance attributes (i.e., name, gender, religion, location etc.,) from different social media accounts associated with a particular twitter user. Although the proposed tool captures complete details of a social media user, yet it neither provides a provenance path nor a propagation history and updates of a digital content published on a social media. Similarly, a provenance path algorithm (Gundecha et al. 2013a) is proposed to capture provenance path of an information to explain how this information propagates on social media but proposed algorithm is not scalable. A log-based provenance graph generation approach (DeBoer et al. 2013) is introduced to identify malicious node in a distributed network but this approach is deemed fit for a static network, not for dynamic network where provenance graphs are frequently changed during execution. Secondly, the logs have an ad hoc structure, not readily available for effectively querying and may not capture complete provenance information. To reconstruct and integrate provenance of messages in social media, a workflow provenance model PROV-SAID (Taxidou et al. 2015, 2018; De Nies et al. 2015) based on W3C PROV data model is proposed. Although the proposed solution identifies the posted tweets that are copied from other published tweets without giving credit to original tweeter like a retweet, but it is suitable for a small dataset only. A trust model (Zhao et al. 2016) is also introduced to assess trustworthiness of both tweet and the user who posted that tweet related to a specific event in Twitter's network. A python library toolkit (Filgueira et al. 2015) is proposed to capture provenance for workflows that collect provenance about every process in the workflow, but not suitable to capture provenance at fine-grained level, i.e., how an element has been generated in the result set etc. A provenance framework based on algebraic structure of semirings (Ramusat et al. 2018) is presented to compute provenance of regular path queries (RPQ) over graph database by applying annotations like labels and weight functions which is a quite complex process. A provenance model (Papavasileiou et al. 2019) for vertex centric graph computation and a declarative data-log based query language is presented to capture and query graph analytics provenance for both online and offline mode. In some other way, a Q-Chase-based algorithm (Namaki et al. 2019) is introduced for efficient implementation of a query chasing process and to compute query rewrite.

A qualitative analysis of existing provenance solutions and our proposed SDP framework based on an evaluation matrix including level of provenance granularity, type of queries supported for provenance generation, provenance visualization, and its applicability is shown in Table 1.

It is obvious from available literature that most of the existing approaches for provenance in graph databases are not scalable to track provenance metadata for social media efficiently. Existing frameworks primarily focused on workflow provenance that captures coarse-grained provenance only rather than detailed fine-grained provenance information. Although, a few frameworks support data provenance that capture fine-grained provenance information, but they do not support all type of queries for provenance capture. To the best of our knowledge, none of the existing frameworks support provenance for historical queries and provenance for data updates. Therefore, the viability of such a framework has become the necessity to engender the trust among social media's users. To address these challenges, we propose a social data provenance framework that efficiently captures provenance data for all types of queries including historical queries and generates a provenance graph database for further analysis.

3 Problem formulation

Twitter's social network graph (Angles and Gutierrez 2018; O'Reilly and Milstein 2011; Aryono 2016), is generally represented by a directed graph $G(V, E)$ with multivariate properties, where V is a set of labeled nodes known as 'Entity', E is a set of directed labeled edges known as 'Relationship', and both V and E can also have their own properties in a form of key-value pair. The source of information is represented by a node 'v' and flow of information is represented by an edge 'e', where $v \in V$ and $e \in E$. In other words, we can say that V is the set of twitter users say $v_1, v_2, v_3, \ldots, v_n$, that produces the information and E is the set of relationships among them. A snapshot of Twitter's social network graph is shown in Fig. 1, where nodes are labeled with USER (Blue Nodes), COUNTRY (Red Nodes), TWEET (Pink Nodes), HASHTAG (Purple Nodes), URL (Yellow Nodes) etc. In a similar fashion, the directed edges represent the flow of information and are labeled with TAGGED, MENTIONED, REPLY_TO, POSTED, FROM, QUOTED etc.

In this paper, Our main goal is to design a zero-information loss graph database (ZILGDB) to maintain the history of insert, delete, and update operations in a form of provenance information. Another goal is to capture provenance for select, aggregate and historical queries, and to generate provenance paths of all the result tuples. Finally, to store captured provenance paths in a provenance graph database (PGDB) for further analysis. To achieve these goals, we state following three main problems:

Problem 1 (*Zero-information loss graph database (ZILGDB) design*) ZILGDB design is further divided into following three sub-problems according to the type of

Table 1 Qualitative analysis of different provenance solutions for social data

Provenance model	Provenance granularity	Provenance capture						Provenance visualization			Application domain
		Select query	Aggregate query	Historical query	Update query	Insert query	Delete query	Provenance querying	Justifying query results	Historical data	
SFM (Kerchner et al. 2016)	Fine-grained level	Yes	No	No	No	No	No	Yes	Limited	No	Generic
Web-based tool (Gundecha et al. 2013b; Ranganath et al. 2013)	Fine-grained level	Yes	No	No	No	No	No	Yes	Limited	No	Generic
Seeking provenance paths (Gundecha et al. 2013a)	Fine-grained level	Yes	No	No	No	No	No	Yes	Yes	No	Generic
RAMP (Park et al. 2011)	Workflow level	Yes	Yes	No	No	No	No	Yes	Yes	No	Generic
Hadoop-Prov (Akoush et al. 2013)	Workflow level	Yes	Yes	No	No	No	No	Yes	Yes	No	Generic
Substructure mining (DeBoer et al. 2013)	Log based	Yes	No	No	No	No	No	Yes	No	No	Application specific
PROV-SAID (Taxidou et al. 2015, 2018; De Nies et al. 2015)	Workflow level	Yes	No	No	No	No	No	Yes	Yes	No	Application specific
Semiring provenance (Ramusat et al. 2018)	Fine-grained level	Yes	No	No	No	No	No	Yes	Yes	No	Limited for three graph algorithms
Ariadane (Papavasileiou et al. 2019)	Fine-grained level	Yes	No	No	No	No	No	Yes (online and offline)	Yes	No	Vertex centric analysis
Proposed SDP	Fine-grained level	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Generic

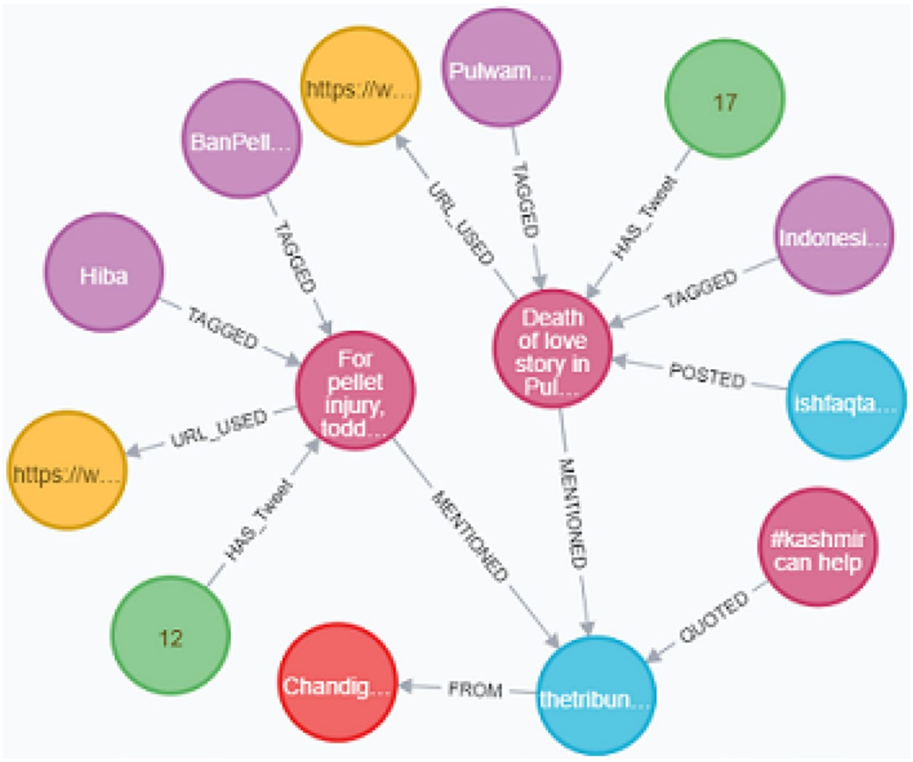


Fig. 1 Twitter's social network graph

operation performed on database, i.e., insert, delete or update operation.

1. Given a Graph $G(V, E)$ and an Update Query Q_U to be executed on G. Our objectives are:
   (a) To find v, a_v, $G_v(v_a, e_a)$ where $v \in V$ is the vertex to be updated, a_v is the attribute of v to be updated, and $G_v(v_a, e_a)$ is a subgraph of G such that $v_a \in V$, $e_a \in E$ where v_a is set of vertices associated with v via edges e_a in G.
   (b) To create a new node as a clone of vertex v in Graph G, and associating it with all v_a via edges e_a same as the original vertex v, but with the updated value of attribute a_v. Afterwards, setting "*valid_from*" attribute of new node as "*current date/time*", and "*valid_to*" attribute of old node v as "*current date/time*".
2. Given a Graph $G(V, E)$ and Insert Query Q_I to be executed on G, our objective is to insert a new node and set the time of its existence, i.e., "*valid_from*" attribute as "*current date/time*".
3. Given a Graph $G(V, E)$ and Delete Query Q_D to be executed on G, our objectives are:
   (a) To retrieve the node v to be deleted as per query Q_D.
   (b) To update the node v with its time of expiry, i.e., "*valid_to*" attribute as "*current date/time*".

Problem 2 (*Generation of provenance paths (P) for all result tuples* (T_n)) Given a Graph $G(V, E)$ and a cypher query (Q), here our objectives are:

1. To generate a query graph G_Q, by executing Q on G, where G_Q is a subgraph of G that comprise of all source nodes contributed towards generation of all result tuples T_n.
2. For every result tuple $t \in T_n$, perform DFS on query graph G_Q to generate provenance paths P of t and insert that in final provenance graph $G_p(V_p, E_p)$.

Problem 3 (*Querying provenance on PGDB for further analysis*) Consider a Provenance Graph $G_p(V_p, E_p)$, our objectives are:

1. To find a subgraph $G_{ps}(v_{ps}, e_{ps})$ of graph G_p such that $v_{ps} \in V_p, e_{ps} \in E_p$ which is contributing directly or indirectly to a specific result tuple for various purposes like audit trail, tracing the source of a result tuple etc.
2. To extend the cypher query by including following user-defined constructs viz. "*All*" , "*Instance*", "*valid_on now*", and "*valid_on 'date'*" for historical data queries.

4 Proposed provenance framework

We propose a social data provenance framework (Rani et al. 2021) build upon zero-information loss graph database (ZILGDB) that efficiently captures provenance for all queries including select, aggregate, update, insert, delete along with historical queries, and generates a provenance graph for further visualizations. ZILGDB is developed by using the concept of Zero-Information Loss Database (Bhargava and Gadia 1993; Rani et al. 2015, 2016). The major steps involved in designing the proposed provenance framework are as follows:

1. To design a data model in Neo4j graph database for social media data with efficient range query support.
2. Designing ZILGDB with data version support to maintain all insert, delete, and update operations in the form of provenance data, that will aids in Historical data queries, Historical queries, and Querying through time.
3. Proposing a provenance generation algorithm to generate provenance data for select and aggregate queries, and to store captured provenance in provenance graph database (PGDB).
4. To provide provenance visualization for various applications like audit-trail, fact investigations etc.

In addition, a performance analysis of provenance capturing and provenance querying is also presented for different query sets. Although, all the queries of different query sets including queries for provenance capturing as well as queries for querying provenance can be directly implemented on top of Neo4J graph database as Neo4j cypher query language is used to write a query statement yet, it could not be efficient for Historical Data Queries. Advantages of our proposed framework as compared to using Neo4j directly for analysis queries:

1. The framework has been developed around the concept of a zero information loss database (ZILD) (Bhargava and Gadia 1993). By a ZILD, we mean that no data value, no user, and no query and its result (seen at the time query was issued) is ever lost. It is very useful in tracking any “data manipulations” that have taken place in an organization.
2. ZILGDB is a special kind of temporal database that provides data version support to maintain the history of all updates in form of provenance data along with the provenance of all insert and delete operations.
3. It supports to perform historical data queries. Historical data query is defined as the query to know all updates/versions of a data object within any specific time range, or instance of a data object at any particular time. These queries cannot be answered directly on Neo4j without our proposed provenance framework.
4. It also assists to capture provenance for historical queries. Historical query is defined as the query that was executed in the past and generates the same result (as seen in the past) in every subsequent execution.
5. To support provenance querying for historical data, we designed following four User-Defined cypher query constructs (UDCs), viz. “instance”, “all”, “valid_on now”, and “valid_on date” in our proposed framework. But in case of using direct Neo4j cypher query, such query constructs are not provided by Neo4j.

4.1 Data model design using graph database

In a graph database, real-world entities are represented as a node. A node can have attributes in the form of a key-value pair and can be associated with other nodes via edges/relationships. Bidirectional relationship between two nodes can be modeled as two separate directed relationships, one in each direction. Nodes and edges with analogues attributes can be grouped under one label respectively.

In the proposed framework, we initially design the data model for Twitter data based on frequently used entities and relationships used in Twitter data analytics. Following entity types, i.e., USER, COUNTRY, TWEET, HASHTAG, URL etc., and relationship types, i.e., FROM, POSTED, TAGGED, MENTIONED, QUOTED, URL_USED, REPLY_TO etc, are used in Twitter data model design. Like a labelled graph, all these entities and relationships are modeled as labeled nodes and edges respectively, and the nodes and edges having similar properties are grouped under one label. Similarly, in accordance with a property graph, each of the nodes and edges have their properties/attributes in the form of a key-value pair. In this way, the extended graph model is described in the following manner:

V: Set of Nodes
E: Set of Edges
TN: Set of Node Types (TWEET, USER, COUNTRY, HASHTAG, URL)
f(TN → V): Function to assign Label to a Node
TE: Set of Edge Type (FROM, POSTED, MENTIONED,QUOTED,TAGGED,URL_USED)
g(TE → E): Function to assign Label to an Edge
AN: Set of Node Attributes/Properties
AE: Set of Edge Attributes/Properties

In accordance with the extended graph model, our Twitter data model design is shown in Fig. 2, which also includes a Timeline *(Year)-[:Has_Month]->(Month)-[:Has_Day]->(Day)-[:HAS_TWEET]->(Tweet)* where an *Year* is associated with all *Months* of year using *Has_Month* relationship

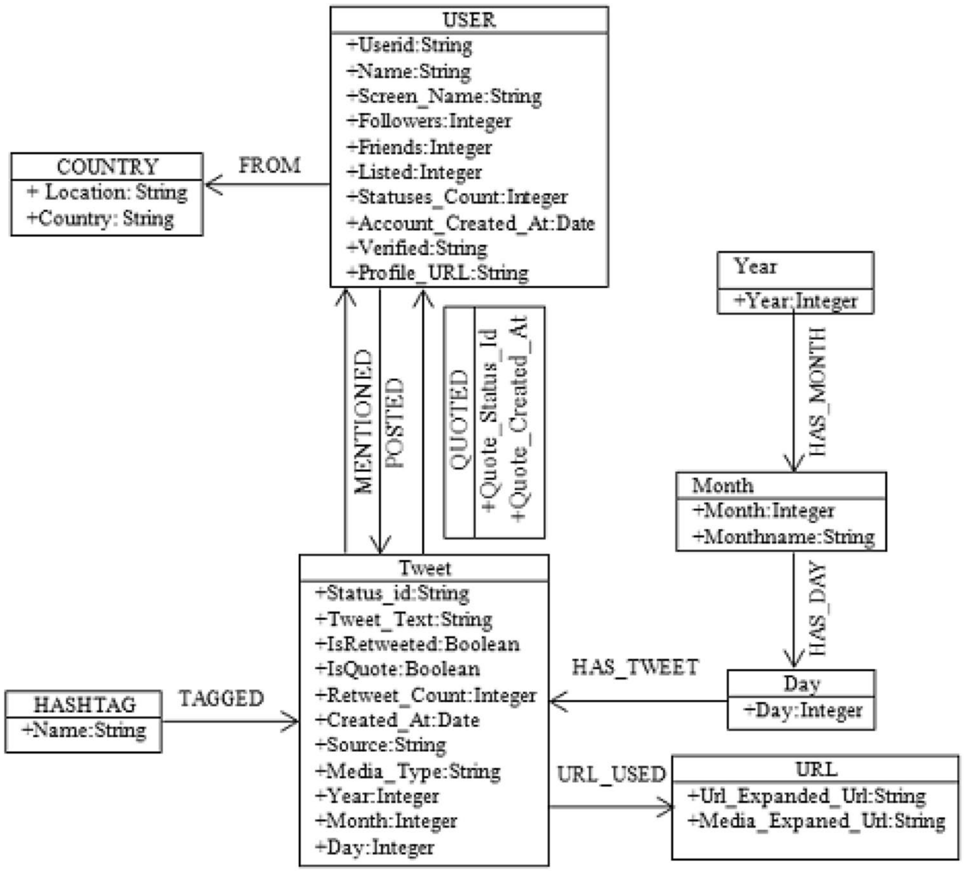


Fig. 2 Twitter data model

which are further associated with all *Days* of month via *Has_Day* Relationship. The *Day* node is associated with *Tweet* node via *HAS_TWEET* relationship, for all the tweets created on that particular day. Further, each *Day* node is associated to next day using *NEXT* relationship as shown in Fig. 3. This timeline is immensely favourable in querying especially for range queries (Robinson et al. 2015) such as tweets posted between any two specific days, i.e.,

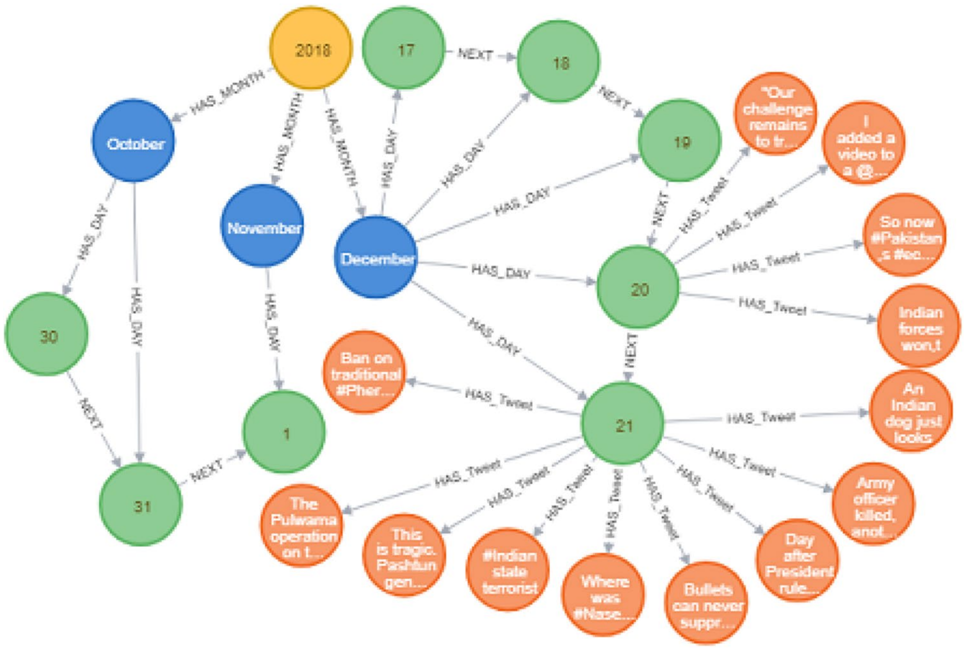

2018
October
November
December
30
31
1
17
18
19
20
21
HAS_MONTH
HAS_DAY
NEXT
HAS_Tweet
"Our challenge remains to tr...
I added a video to a @...
So now #Pakistan s #ec...
Indian forces won,t
An Indian dog just looks
Army officer killed, anot...
Day after President rule...
Bullets can never suppr...
Where was #Nase...
#Indian state terrorist
This is tragic. Pashtun gen...
The Pulwama operation on t...
Ban on traditional #Pher...

Fig. 3 Timeline with Tweet

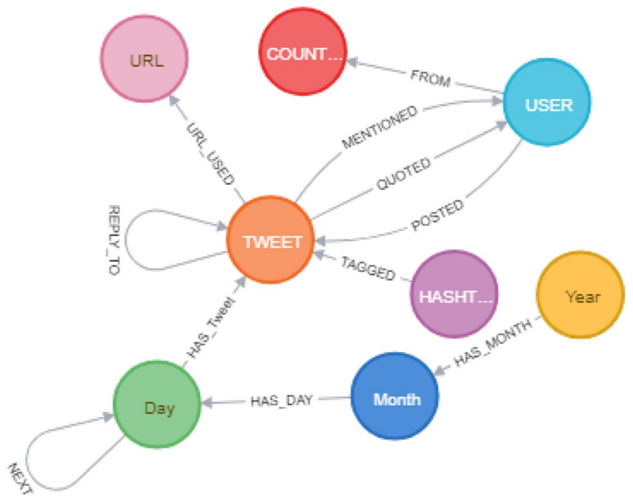


Fig. 4 Neo4j schema for Twitter data set

16/12/2018 to 18/12/2018 as shown in example query 1. To retrieve all the tweets posted between two specific days, a user just needs to specify "*start date*" and can traverse all the days up to "*last date*" via NEXT relationship, instead of searching all the tweets in database. Consequently, all the tweets related to these dates are retrieved. The Neo4j Schema of Twitter data set for efficient queries using timeline is shown in Fig. 4.

Example query 1: Retrieve all the tweets posted by a specific user (Screen_Name : KashmirCause_) between 16/12/2018 and 18/12/2018.

With Timeline: MATCH (y:Year {year: 2018})-[hm:HAS_MONTH]->(m:Month {month: 12})-[hd: HAS_DAY]->(d:Day {day: 16})-[:NEXT*0..2]->(day) with day MATCH (day)-[:HAS_Tweet]->(t: TWEET) <-[:POSTED]-(u:USER {Screen_Name: 'Kashmir-Cause_'}) return *

Without Timeline: MATCH (t:TWEET)<-[:POSTED]-(u:USER {Screen_Name:'KashmirCause_'}) where t.year=2018 and t.month=12 and t.day>=16 and t.day<=18 return *

4.2 ZILGDB architecture

The schematic diagram of proposed ZILGDB architecture is shown in Fig. 5. The architecture consists of following main components viz., Query Parser, Query Rewriter, Query Generator, and Graph Database. When a user issues a query in step 1, it is first sent to the Query Parser (QP), where query parser parses the query and also identifies the type of that query, i.e., Insert (I), Update (U), or Delete (D) query. If the issued query type is an "*Insert Query*" then the parsed results are sent to the Query Rewriter (QR) as mentioned in step I_1 and corresponding rewritten query (Q_I) is generated in step I_2. Here, the "*valid_from*" attribute of this new node is being set as a "*current date/time*" and then it is sent to the graph database for further execution. Now, if the issued query type is a "*Delete Query*" then the parsed results are sent to the Query Generator (QG) and the corresponding update query (Q_U) is generated by setting "*current date/time*" as the value of "*valid_to*" attribute of the node to be deleted as shown in step D_1 and D_2 respectively, and afterwards, it is sent to the graph database for further execution. Similarly, if the issued query type is an "*Update Query*" then the parsed results are sent to both QG and QR as mentioned in step U_{1a} and U_{1b} respectively. Then, in step U_2, corresponding select query (Q_S) is generated by QG and sent for the execution on graph database, to retrieve the original node 'v' to be updated. After retrieving the node to be updated, in step U_3, the corresponding create query (Q_C) is generated by QG to create a new node 'v_c' as a clone of 'v' and it is further associated with all the nodes as 'v' via same type of edges, but with updated value of attribute as specified in original query. Finally, Q_C and a rewritten update query (Q_{RU}) which is generated by QR, in steps U_4 and U_5, respectively, are sent for the execution on graph database to maintain the history of update operation.

Algorithm 1 ***ZILGDB Design***: Design ZILGDB (Zero Information Loss Graph Database)

```
Require: Graph G(V, E), Query Q (Insert, Update, or Delete Query)
Ensure: Graph G(V, E) with history maintained
1:  Parsed Result (P), Query Type ($Q_T$) ← Query Parser (Q)
2:  if $Q_T$ is Update-Query then
3:      Obtain $A_i$ and $v_n$ ← P // where $A_i$ is the attribute to be updated, $v_n$ is new value of attribute $A_i$
4:      Get Select Query($Q_S$) corresponding to Q, which will return the node needs to be updated
5:      Vertex $V_o$ ← Execute (G, $Q_S$) // where $V_o \in V$, is a node to be updated
6:      Vertex $V_u$ ← Clone $V_o$
7:      Add Edges to $V_u$ ← Clone Edges of $V_o$
8:      Updated $V_u$ ← Update $A_i$ with $v_n$ of $V_u$
9:      Add Attribute "valid_to" to $V_o$ with value as "current date/time"
10:     Add Attribute "valid_from" to $V_u$ with value as "current date/time"
11: else if $Q_T$ is Insert-Query then
12:     $Q_I$ = Get Rewritten Insert Query by adding "valid_from" attribute with value as "current date/time"
13:     Execute (G, $Q_I$)
14: else
15:     $Q_U$ = Get Corresponding Update Query by adding "valid_to" attribute with value as "current date/time"
16:     Execute (G, $Q_U$)
17: end if
```

Pseudo-code of ZILGDB design is mentioned in Algorithm 1. A "*valid_from*" attribute is added to every node whenever it is inserted (refer to lines 11 to 13). This attribute stores node's created time, i.e., "*current date/time*" as a value. Whenever a delete operation is performed, a "*valid_to*" attribute is then added to the corresponding node with "*current date/time*" as value (refer to lines 15 and 16). It shows that the node has already expired but still exists in the database, which help in past queries execution to obtain the same result as their previous executions before delete operation occurs. Whenever an update operation is performed, initially the original node which is to be updated is retrieved using generated select Query Q_s from parsed results (P) of issued update query (refer to lines 3 to 5). Afterwards, original node is cloned (Updated Cloned Node), but with an updated value of that attribute as per the issued update query, and consequently its "*valid_from*" attribute is set to the "*current date/time*" as a value. Updated cloned node is assigned the same labels, and is associated with all the nodes with same type of edges as per its corresponding

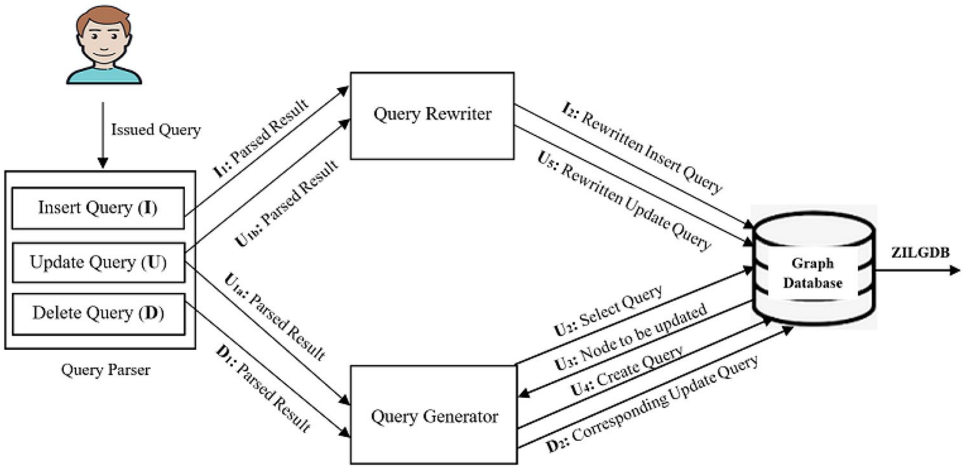


Fig. 5 ZILGDB architecture

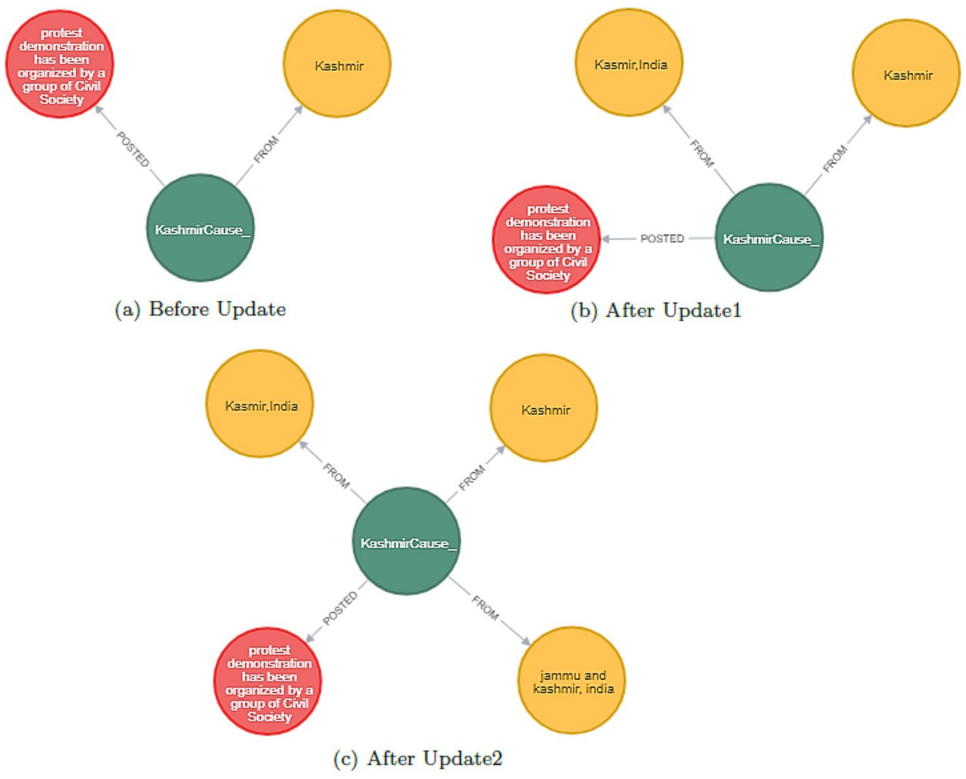
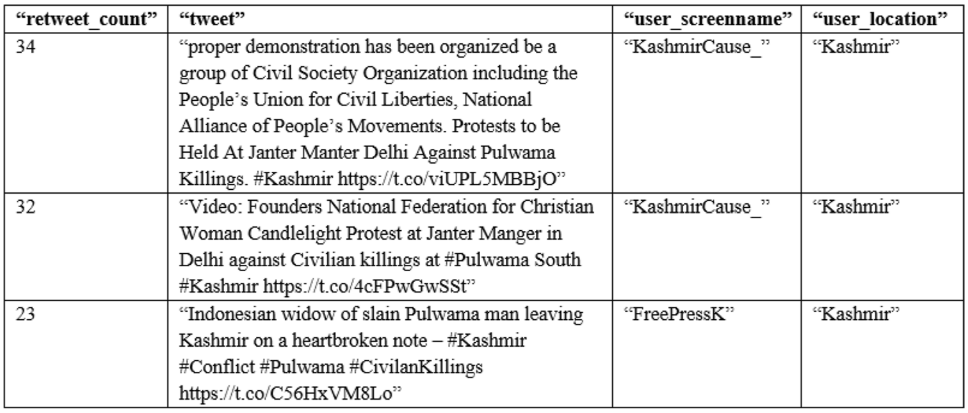


Fig. 6 Zero-loss graph database

original node (refer to lines 6 to 9). Finally, the "*valid_to*" attribute is also added to the original node that stores "*current date/time*" as its value, i.e., expire time (refer to line 10). For example, Fig. 6 shows a snapshot of ZILGDB, where a USER named "KashmirCause_" is from Location "Kashmir" since 15-08-2019, as shown in Fig. 6 (see (a) Before Update). User updates its location from "Kashmir" to "Kasmir, India"; Now, the original node is cloned with all its attributes but the location attribute of cloned node is updated with value "Kasmir, India". The "*valid_to*" attribute of original node is set to the "*current date/time*" and "*valid_from*" attribute of cloned node is set to the "*current date/time*" as a value, as shown in Fig. 6 (see (b) After Update1). After this, user again updates its location from "Kasmir, India" to "Jammu and Kashmir, india" as shown in Fig. 6 (see (c) After Update2) in the same manner as earlier shown in Fig. 6 (see (b) After Update1).

4.3 Provenance generation and storage

Proposed SDP framework supports generation of provenance paths for all result tuples of select, aggregate and historical queries. The captured provenance path is also stored in provenance graph database (PGDB) for visualizations and analysis.



"retweet_count"	"tweet"	"user_screenname"	"user_location"
34	"proper demonstration has been organized be a group of Civil Society Organization including the People's Union for Civil Liberties, National Alliance of People's Movements. Protests to be Held At Janter Manter Delhi Against Pulwama Killings. #Kashmir https://t.co/viUPL5MBBjO"	"KashmirCause_"	"Kashmir"
32	"Video: Founders National Federation for Christian Woman Candlelight Protest at Janter Manger in Delhi against Civilian killings at #Pulwama South #Kashmir https://t.co/4cFPwGwSSt"	"KashmirCause_"	"Kashmir"
23	"Indonesian widow of slain Pulwama man leaving Kashmir on a heartbroken note – #Kashmir #Conflict #Pulwama #CivilanKillings https://t.co/C56HxVM8Lo"	"FreePressK"	"Kashmir"

Fig. 7 Snapshot of result of example query 2

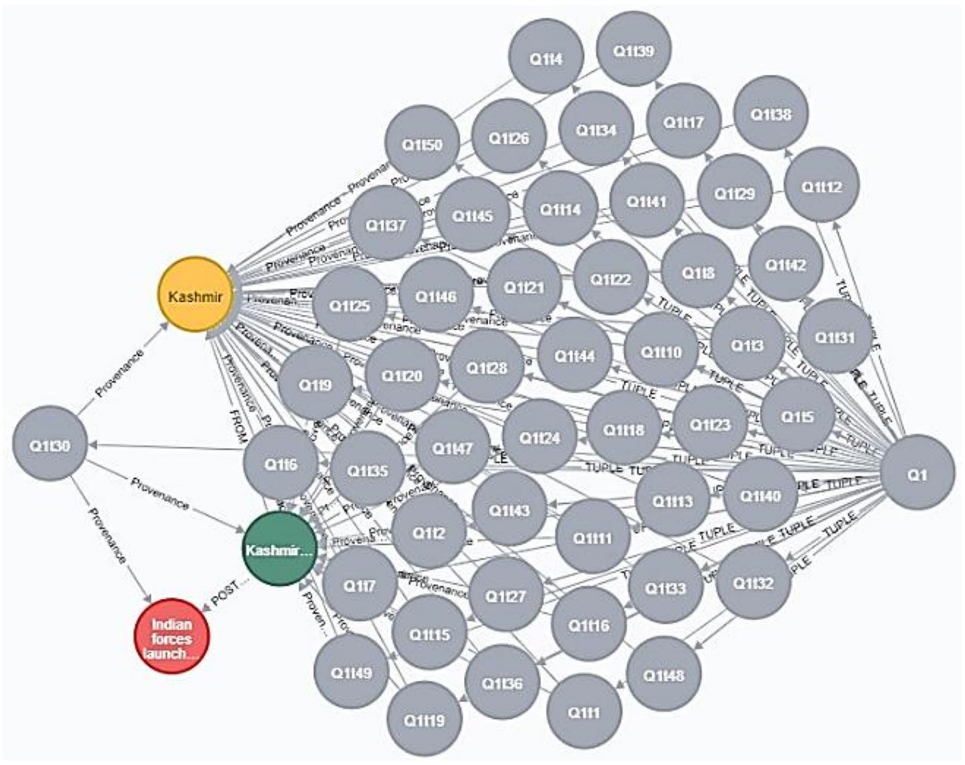


Fig. 8 Partial provenance graph of example query 2

Algorithm 2 ***ProvGraph Generation***: Extraction of Provenance Graph for a Result Tuple of Query Q on Graph G)

```
Require: Data Graph G(V, E) and Cypher Query Q
Ensure: Provenance Graph $G_P(V_P, E_P)$ of Query Q on Graph G where $V_P \in V$ and $E_P \in E$
1: Obtain all Result Tuples $T_n$, Query Graph $G_Q(V_Q,E_Q) \leftarrow$ Execute (G,Q)
   //Where $V_Q \in V$ and $E_Q \in E$ and For each $T_i \in T_n$, $T_i = (A_1, A_2, \ldots, A_n)$, where $A_i$ = Property of a Node or an Aggregate Function
2: for all $T_i \in T_n$ do
3:     Select $A_i$  //$A_i$ should be Property of a Node not an Aggregate Function
4:     $V_t \leftarrow$ Extract Vertex $V_i$ of $A_i$ in $G_Q$
5:     Provenance_Path (P) $\leftarrow$ DFS ($G_Q$ ,$V_t$)
         //Apply DFS on $V_t$, P is set of all provenance paths=$p_1,p_2,\ldots,p_n$, Each $p_i \in P$ is a sequence of $V_d \in V_Q$ and $E_d \in E_Q$
6:     $G_P(V_P, E_P) \leftarrow$ Insert P        //$G_P$ is Provenance Graph
7: end for
8: Return $G_P(V_P, E_P)$
```

Algorithm 2 shows high-level details of provenance graph generation that accepts graph $G(V, E)$ and *cypher query Q (select or aggregate)* as inputs and returns provenance graph of all result tuples T of Q as output. Initially, the cypher query Q is executed on Graph G which returns all the result tuples as well as *Query Graph* (G_Q) containing all the nodes and associated edges contributed to all result tuples of Q, as complete provenance graph of Q (refer to line 1). After that, for some/all result tuples, corresponding vertex is extracted from *Query Graph* (G_Q) (refer to lines 2 to 4). Then, DFS is applied on extracted vertex to generate provenance paths of result tuple (refer to line 5). Provenance paths are then added to provenance graph database (PGDB) for further analysis,

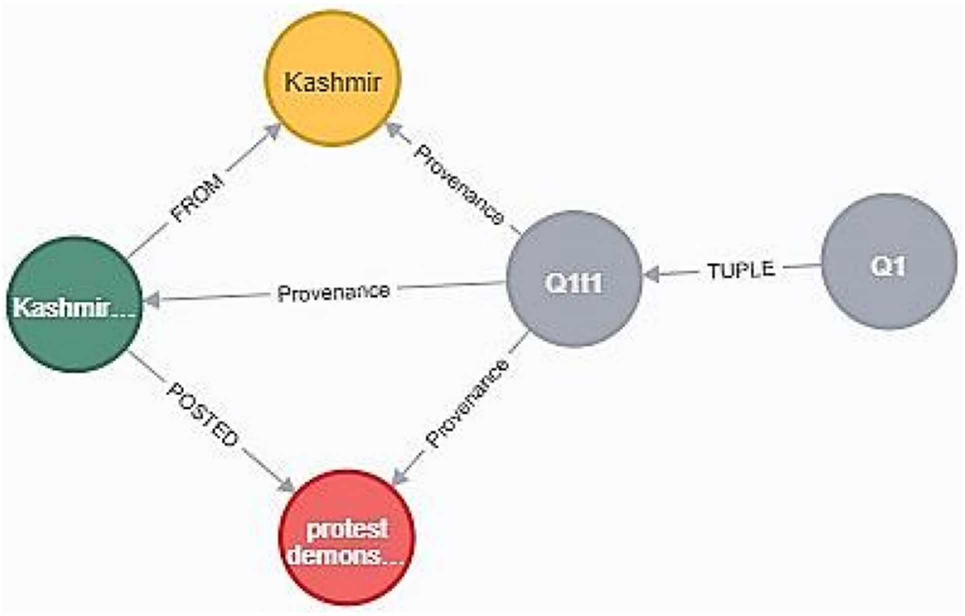


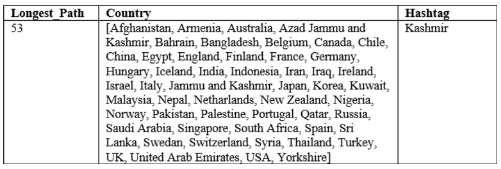
Fig. 9 Provenance of result Tuple t1 of example query 2

i.e., provenance visualization (refer to line 6). Illustrative example queries of proposed algorithm are presented below:

Example query 2: Find top 50 most retweeted tweets by any user from a specific location "Kashmir" in a given range of days with the retweet count, tweet text, user's name, in descending order of the retweet count.

Cypher query 2: MATCH (c:COUNTRY)<-[f:FROM]-(u:USER)-[p:POSTED]->(t:TWEET)<-[ht:HAS_Tweet]-(d:Day)<-[hd:HAS_DAY]-(m: Month)<-[hm:HAS_MONTH]-(y:Year) where c.Location ='Kashmir' and y.year=2018 and m.month=12 and d.day>=18 and d.day<=20 return t.Retweet_Count AS retweet_count, t.Tweet_Text as tweet, u.Screen_Name as user_ screenname, c.Location as user_location order by t.Retweet_Count desc limit 50

Snapshot of example query 2 result is shown in Fig. 7. Here, first row shows most retweeted tweet posted by user named "KashmirCause_" from location "Kashmir" and so on for other rows in descending order of retweet_count. To explore why this result has appeared in result set, and how it is derived; we need to know about provenance information for the query result set. Figure 8 shows partial provenance graph of all result tuples (labeled as '*QUERYTUPLE*'), i.e., Q1t1, Q1t2 etc. of query Q1 (labeled as '*QUERY*'). Further, each result tuple node is associated with all source nodes contributed towards it via '*provenance*' relationship. Figure 9 shows provenance graph of result tuple 1, i.e., Q1t1 of Q1. Provenance graph also stores the complete information about query which has been executed, i.e., issued query, time of execution as the attributes of '*QUERY*' node.



Longest_Path	Country	Hashtag
53	[Afghanistan, Armenia, Australia, Azad Jammu and Kashmir, Bahrain, Bangladesh, Belgium, Canada, Chile, China, Egypt, England, Finland, France, Germany, Hungary, Iceland, India, Indonesia, Iran, Iraq, Ireland, Israel, Italy, Jammu and Kashmir, Japan, Korea, Kuwait, Malaysia, Nepal, Netharlands, New Zealand, Nigeria, Norway, Pakistan, Palestine, Portugal, Qatar, Russia, Saudi Arabia, Singapore, South Africa, Spain, Sri Lanka, Swedan, Switzerland, Syria, Thailand, Turkey, UK, United Arab Emirates, USA, Yorkshire]	Kashmir

Fig. 10 Result of example query 3

Example query 3: Find topmost hashtag which appeared in most number of countries along with the number and list of the distinct countries it appeared in.

Cypher query 3: MATCH (h:HASHTAG)-[tg:TAGGED]->(t:TWEET) <-[p:POSTED]-(u:U_ SER)-[f:FROM]->(c:COUNTRY) return (count(distinct (c.Country))) AS Longest_Path, collect(distinct (c.Country)) as Country, h.Hashtagname AS Hashtag limit 1

The proposed framework is also capable to capture provenance for aggregate queries, as given in above example query 3. The query result of this aggregate query describes that users from 53 different countries are using a common most popular hashtag "Kashmir" in their posted tweets as shown in Fig. 10. The provenance graph of this aggregate query is shown in Fig. 11, which can be analysed further for justifying query results such as why does the country name "Australia" is appeared in the result set? how many users from this country are using the most popular hashtag in their posted tweets? etc.

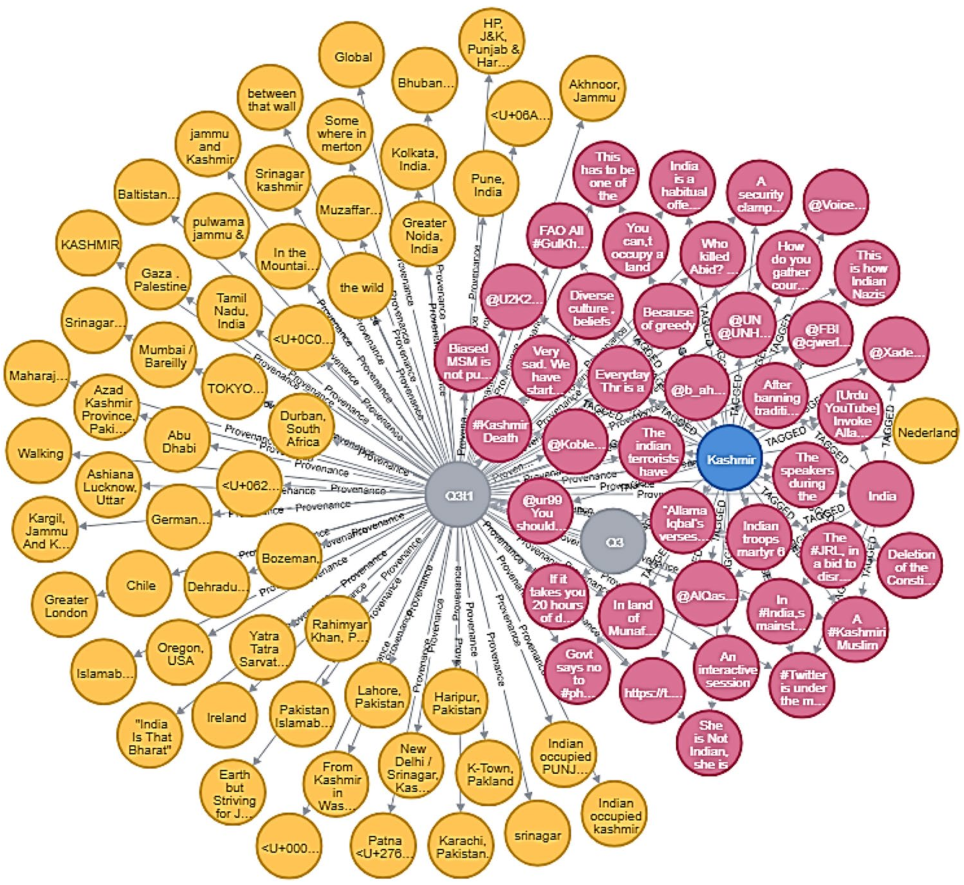


Fig. 11 Snapshot of provenance of example query 3

4.4 Querying provenance

Provenance of a query result provides the details of all the relevant paths originating from the source graph which contributed to generate the query result as explained in previous Sect. 4.3. Querying provenance information aids in explaining and justifying the result of a data retrieval query through forward or backward tracing. It is helpful in social data analytics where the accountability of an analysis is largely relying on the data quality and trustworthiness of input data. Also, it has the ability to trace out the root cause in a social data analytics. Provenance enabled social media can offer an effective approach to address increasingly challenging issue of trust in social media. In this section, we explain the querying provenance for historical data query and to justify the query results with some example queries.

Example provenance query PQ1: Let's consider the result set and partial provenance graph of Example Query 2 shown in Figs. 7 and 8 respectively. We now explain the derivation of result tuple t2 in the result set.

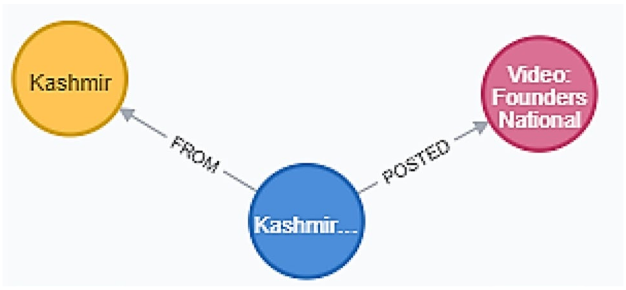


Fig. 12 Result of PQ1

Cypher query PQ1: *MATCH (n:QUERY)-[t:TUPLE]->(n1: QUERYTUPLE)-[p:Provenance]->(n2) where n.qid='Q1' and n1.qtid='Q1t2' RETURN n2*

After executing the above cypher query PQ1, we obtain the provenance path of result tuple t2 as shown in Fig. 12. This provenance path shows the three nodes from source graph contributing towards generation of t2. Relationships among these nodes explain '*How*' they are contributing towards generation of result tuple.

Proposed framework also supports multi-depth provenance querying to determine the direct and indirect sources

Table 2 Sample data retrieval queries

Query No.	Query
Q1	Find top *N* most retweeted tweets by the users from a given location in a given range of days with the retweet count, tweet text, user's name, in descending order of the retweet count
Q2	Find top *N* users who used a Hashtag "India" in a tweet with the most number of retweets with user's name, tweet text, and retweet count in descending order of the retweet count
Q3	Find topmost hashtag that appeared in the most number of Countries with the number of countries it appeared in along with the list of the distinct countries it appeared
Q4	Find distinct locations along with the month and the date of a tweet posted with a given hashtag in a given range of days, with tweet text and retweet count in descending order of the retweet count
Q5	Find *N* users who used a given set of hashtags in their tweets with the user's name and the country to which the user belongs in the alphabetical order of the names

of a result tuple. Multi-depth provenance query up to depth 2 is explained by following example provenance query PQ2.

Example provenance query PQ2: Let's consider the result tuple with tuple id Q1t2 shown in Fig. 8, find the direct and indirect sources of Q1t2 up to depth 2.

Cypher query PQ2: MATCH (n:QUERY)-[t:TUPLE]->(n1: QUERYTUPLE)-[*..2]->(a) where n.qid= 'Q1' and n1.qtid= 'Q1t2' return *

Proposed framework also allows querying historical data by introducing four new query constructs viz., "*instance*", "*all*", "*valid_on now*", and "*valid_on 'date'*" as an extended cypher query constructs. It supports querying a data element with a given time in the past and with a time range specified in the query statement. Initially, the user issues a provenance query with extended cypher query constructs which is automatically rewritten and passed to ZILGDB for further execution. The example provenance queries (PQ3, PQ4, PQ5, and PQ6) for querying historical data are explained below:

Example provenance query PQ3: Display the current location of user for the graph shown in Fig. 6.

Extended cypher query PQ3: MATCH instance (u:USER)-[:FROM]->(c:COUNTRY) where n.screen_name= 'KashmirCause_' return c.Location valid_on now

Rewritten cypher query PQ3: MATCH (n:USER)-[:FROM]->(c:COUNTRY) where n.screen_name= 'KashmirCause_' and c.valid_to is null return c.Location

Above query generates current location, i.e., "Jammu and Kashmir, india" in its result set.

Example provenance query PQ4: Display the location of user on 01/10/2019 for the graph shown in Fig. 6.

Extended cypher query PQ4: MATCH instance (u:USER)-[:FROM]->(c:COUNTRY) where n.screen_name= 'KashmirCause_' return c.Location valid_on date('2019-10-01')

Rewritten cypher query PQ4: MATCH (n:USER)-[:FROM]->(c:COUNTRY) where n.screen_name= 'KashmirCause_', date('2019-10-01')>=c.valid_from and c.valid_to>=date ('2019-10-01') return c.Location

Above query generates location, i.e., "Kashmir" in its result set.

Example provenance query PQ5: Display all the location updates of user till now for the graph shown in Fig. 6.

Extended cypher query PQ5: MATCH all (u:USER)-[:FROM]->(c:COUNTRY) where n.screen_name='KashmirCause_' return c.Location valid_on now

Rewritten cypher query PQ5: MATCH (n:USER)-[:FROM]->(c:COUNTRY) where n.screen_name= 'KashmirCause_' return c.Location

Above query generates locations, i.e., "Kashmir", "Kasmir, India", and "Jammu and Kashmir, india" in its result set.

Example provenance query PQ6: Display all the location updates of user valid on 20/11/ 2019 for the graph shown in Fig. 6.

Extended cypher query PQ6: MATCH all (u:USER)-[:FROM]->(c:COUNTRY) where n.screen_name='KashmirCause_' return c.Location valid_on date('2019-11-20')

Rewritten cypher query PQ6: MATCH (n:USER)-[:FROM]->(c:COUNTRY) where n.screen_name ='KashmirCause_' and (date('2019-11-20')>=c.valid_to or (c.valid_from<= date('2019-11-20') and c.valid_to is null)) return c.Location

Above query generates all the locations of user till 20/11/2019, i.e., "Kashmir" and "Kasmir, India" in its result set.

5 Data set and evaluation

All the experiments are performed on a Windows machine with Intel i7-8700 processor @ 3.20 GHz and 16 GB RAM. Neo4j Desktop version 1.2.1 with embedded Neo4j enterprise edition 3.5.6 has been used as a Graph Database. Neo4j extension library APOC (Awesome Procedures On Cypher) version 3.5.0.4 is included to call APOC library procedures with some modifications to import/export the social data set from csv files to Neo4j graph database and vice versa. Java version 8 has been used as front end programming language to interact with Neo4j graph database. To perform experimental analysis, publicly available twitter dataset related to an incidence of 11 killings at Pulwama district of Jammu and Kashmir, India on 15/12/2018 (Kaggle 2018), has been used. By using cypher scripts, experimental dataset is fed into Neo4j graph database, and then ZILGDB is designed as per the data model in Sect. 4.1. Our initial database consists of around 35,000 nodes and 78,000 edges, and grows gradually after performing each data update operation. Provenance graph database (PGDB) is being created to store the captured provenance information for each query executed on ZILGDB. We now present the analysis of proposed framework based on execution overhead in terms of provenance capturing and querying.

5.1 Provenance capture analysis

To perform experimental analysis of provenance capture, we have executed 24 different data retrieval queries on ZILGDB to capture their provenance information using timeline wherever it is required. A sample set of data retrieval queries are shown in Table 2. As we have already explained in Sect. 4.1, through example query 1, that the performance of a range query using timeline is much efficient as compared to a query executed without timeline (refer to Fig. 13). In a range query, the start and end dates are mentioned explicitly in the query statement, therefore with timeline execution, all the tweets those are posted in-between these dates (including both the dates) and linked through "HAS_Tweet" relationships are only required to search instead of searching the whole database. Thus, no any significant change is measured in the query performance after increasing the size of database. But, in case of without timeline execution, we have to search whole database to retrieve all the tweets posted between these two dates. This degrades the performance, with increase in the number of tweets.

Initially, all the data retrieval queries are executed 42 times without any provenance capturing mechanism. Afterwards, the same set of data retrieval queries are executed 42 times with provenance capturing mechanism. To calculate the average execution time of each query, we have dropped the maximum and the minimum execution time, and then taken average of the remaining 40 values. Average execution times of all the queries are given in milliseconds (ms). Because of the couple of order difference in execution times, the execution performances of all the queries without provenance capture and with provenance capture mechanism are shown separately in Figs. 14 and 15, respectively. We found that select queries which are generating larger number of result tuples are taking more time for capturing and storing provenance information as compared to those select queries having lesser number of results tuples in their result set. This is because the proposed framework capture and store provenance information of each result tuple of a query, which increases the execution time with increase in number of result tuples. It can be seen in Fig. 15 that Queries Q2 and Q23 are taking longer execution time as compared to other queries, because the number of result tuples generated by each of them is 100.

The proposed framework also provides provenance capturing support for aggregate queries using aggregate functions such as collect, count, min, and max. Here, query Q3 (an aggregate query) uses collect function to retrieve all the

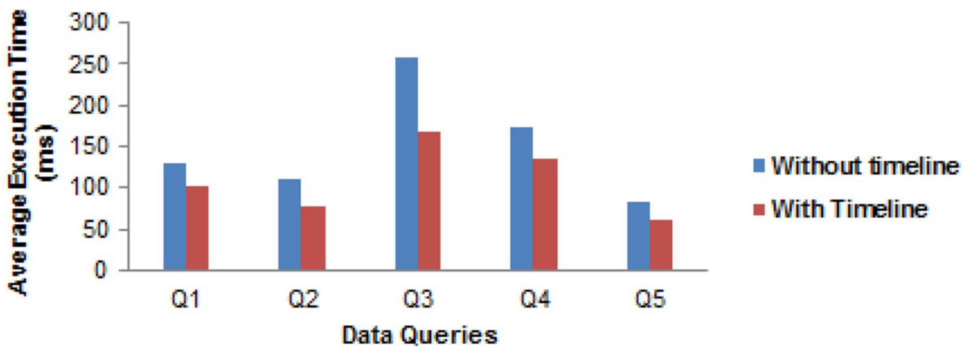


Fig. 13 Query performance with and without timeline

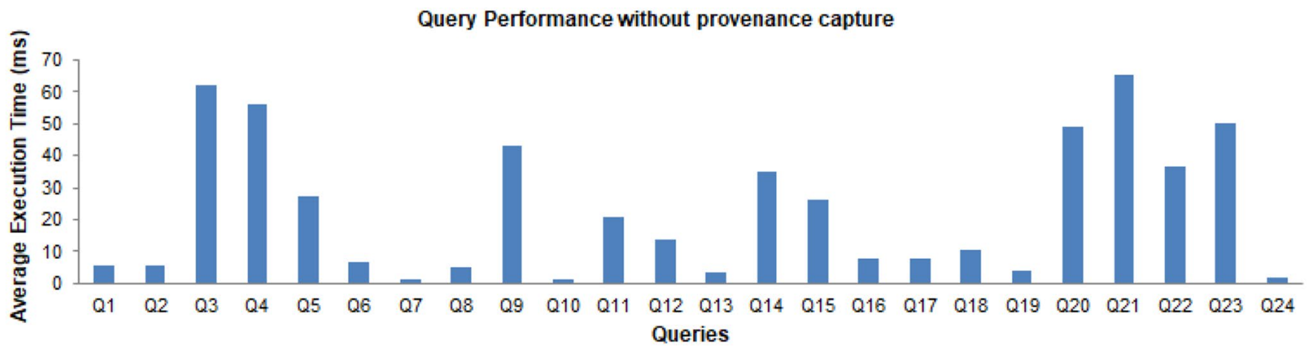


Fig. 14 Query performance without provenance capture

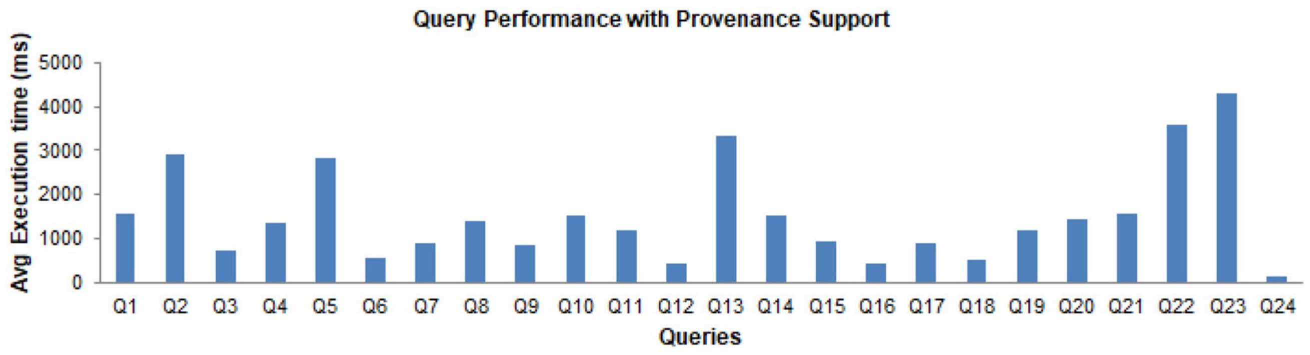


Fig. 15 Query performance with provenance capture

countries which are using topmost hashtag in their tweets and returns only one result tuple in its result set. On the other hand, query Q6, shown in Fig. 15, is not an aggregate query like Q3 and generates 6 result tuples in its result set, still its performance is better than Q3, even though it generates more number of result tuples as compared to Q3. On the other hand, query Q9, shown in Fig. 15, is also an aggregate query generating 10 result tuples and its execution time is comparatively higher than aggregate query Q3 which is generating 1 result tuple. Thus, it can be concluded that aggregate queries are taking longer execution time as compared to select queries. Secondly, execution time of aggregate queries with more number of result tuples is comparatively higher than other aggregate queries with lesser number of result tuples.

This study makes a valuable contribution in the field of Social Data Analytics. In this analysis, we observed that whenever an aggregation is performed on a very large number of input sources, then it is required to capture the details of all the sources that are contributing to derive it, hence it takes longer time as compared to other aggregate queries those are performing aggregation on a smaller number of sources such as query Q3 and Q9 in Fig. 15. In this way, the overall performance of an aggregate query depends on the number of sources contributing towards any result. Therefore, in some cases it may also found that the aggregate queries are performing better than the select queries.

Table 3 Sample data update queries

Query No.	Query
Q1	Update location of user with screen name "KashmirCause_"
Q2	Update name of user with screen name "Kashmir-Cause_"
Q3	Update URL of a user with given screen name
Q4	Insert new post of a user
Q5	Delete a specific post of a user

The proposed framework also supports provenance capturing for insert, delete, and update queries. A sample set of data update queries are shown in Table 3. The following parameters are used to capture the provenance information for update queries, viz. "*Previous value before update*", "*time till the previous value was valid*", and "*time from when new updated value is valid*" as shown in sample graph in Fig. 6. Similarly "*valid_from*" and "*valid_to*" parameters are used in the case of insert and delete query respectively. The provenance information captured for insert, delete, and

Table 4 Sample queries on provenance

Query No.	Provenance query
Q1	Why Hashtag "Kashmir" is appeared in the result set of Query Q3?
Q2	Retrieve all the nodes those have contributed to produce tuple t1 of Query Q1
Q3	Why Hashtag "Kashmir" is arrived as longest path of countries in Query Q3?
Q4	How Tweet t1 has been derived as result in Query Q7?
Q5	Which users from "Jammu and Kashmir" uses top hashtag(Maximum Tagged) in Query Q3?
Q6	How and which user has been contributed to produce result tuple t2 of Query Q2?
Q7	Display the current location of user with screen name "KashmirCause_"
Q8	Display the location of user with screen name "KashmirCause_" on 01/10/2019
Q9	Display all the location updates of user with screen name "KashmirCause_" till now
Q10	Display all the location updates of user with screen name "KashmirCause_" till 11/10/2019

update queries can be used for both historical data queries, and queries executed in the past (historical query) on a specific time.

5.2 Provenance querying analysis

The performance analysis of querying provenance information stored in provenance graph database (PGDB) is presented in this section. A set of 20 provenance queries are executed for analysis of querying provenance. Out of these 20 queries, a sample set of 10 provenance queries are shown in Table 4. This sample set contains both the type of provenance queries, i.e., provenance queries for query results, and provenance queries for historical data. Provenance graph database consists of approximate 22,000 nodes and 43,000 relationships at the time of execution of these provenance queries. Each query in the provenance query set is executed 12 times. To calculate the average execution time of each query, we have dropped the maximum and the minimum execution time, and then taken average of the remaining 10 values. A performance analysis of 10 provenance queries for query results are shown in Fig. 16. The average execution times of all the queries are mentioned in milliseconds (ms).

As we have seen in the Sect. 4.3 that the provenance graph of a query result contains a separate node for each data retrieval query, labelled as "*QUERY*", and their result tuples, labelled as "*QUERYTUPLE*". These "*QUERYTUPLE*" nodes are associated with corresponding "*QUERY*" node via "*TUPLE*" relationship. "*QUERYTUPLE*" nodes are further linked with all the source nodes those are contributing to produce it through a relationship, labelled as "*provenance*" (refer to Figs. 8 and 9). This makes the searching process faster by reducing the search space to a particular portion of the provenance graph rather than the whole graph. As a result, a small execution time is measured in querying provenance for query results. In this way, it is clear from Fig. 16 that the performance of querying provenance for query results is very efficient.

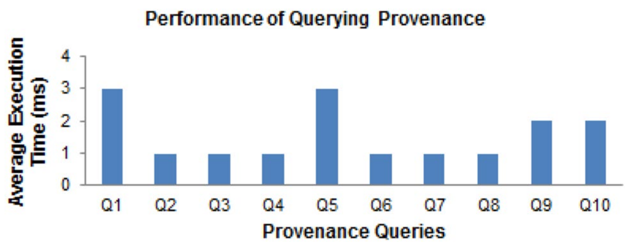


Fig. 16 Querying provenance for justifying result tuples

In case of querying provenance for historical data queries, such as PQ3, PQ4, PQ5, and PQ6 (explained in Sect. 4.4), a longer execution time is measured as compared to querying provenance for query results (refer to Fig. 17). As the query statement is initially written in the extended cypher query using various user-defined constructs such as "*all*", "*instance*", "*valid_on now*", and "*valid_on 'date'*", which is further required to rewrite the statement into the Neo4j cypher query for execution on database. This rewriting process may incur some execution overheads.

5.3 Comparative analysis with existing frameworks

A qualitative analysis of existing provenance solutions with our proposed SDP framework based on an evaluation matrix is already given in Table 1 of Sect. 2. Additionally, this section provides a detail explanation about why the provenance information that are captured by our proposed framework cannot be answered by the other provenance capturing systems.

1. On a social media platform, users frequently update their profiles and posts by adding, removing or changing the information. Therefore, in such application time values are required to tied up with data values to indicate the time interval of their existence. The database that imposes time values with data and levying sequential

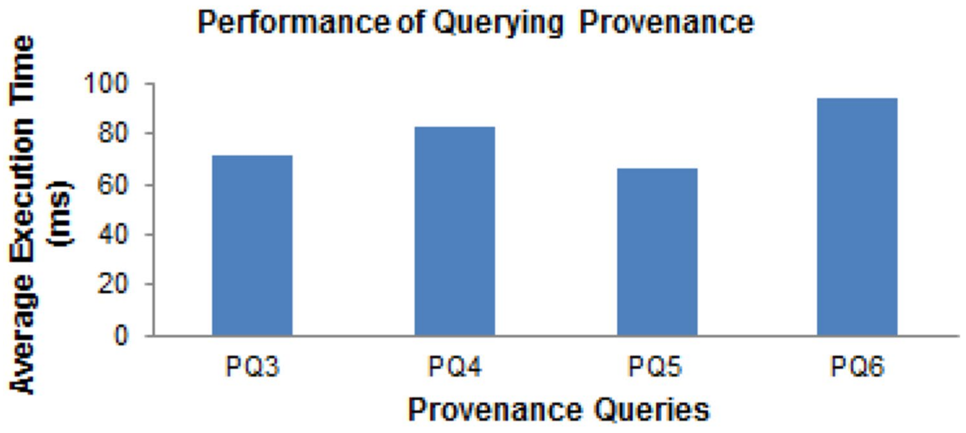


Fig. 17 Querying provenance for historical data

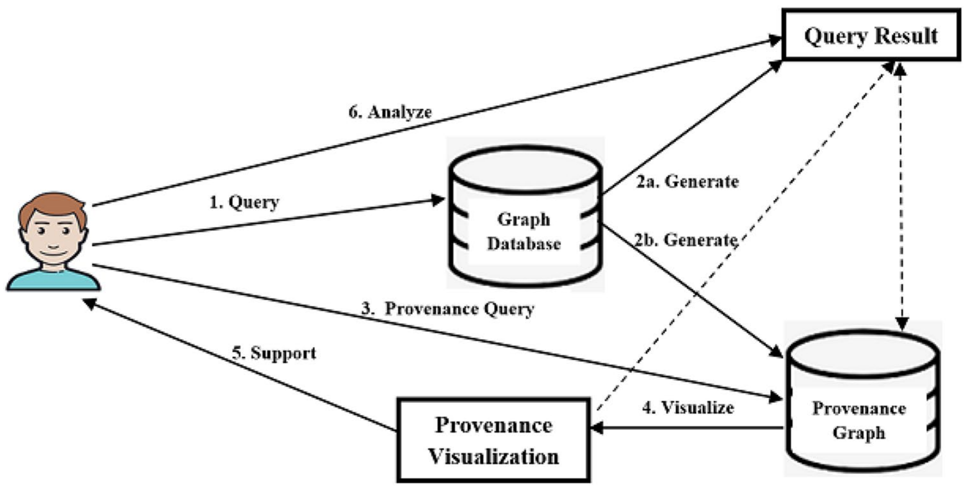
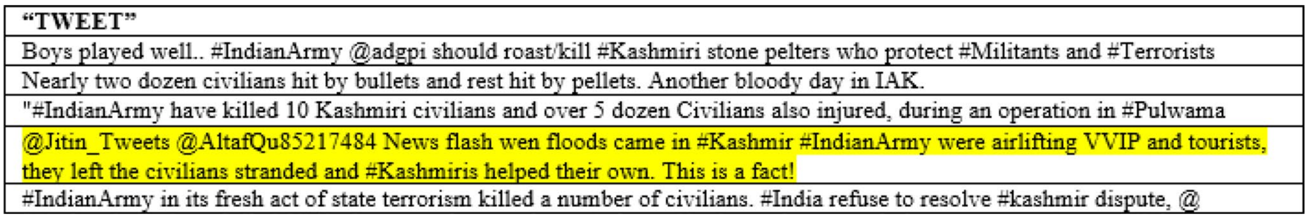


Fig. 18 A cyclic process model based on provenance framework



"TWEET"
Boys played well.. #IndianArmy @adgpi should roast/kill #Kashmiri stone pelters who protect #Militants and #Terrorists
Nearly two dozen civilians hit by bullets and rest hit by pellets. Another bloody day in IAK.
"#IndianArmy have killed 10 Kashmiri civilians and over 5 dozen Civilians also injured, during an operation in #Pulwama
@Jitin_Tweets @AltafQu85217484 News flash wen floods came in #Kashmir #IndianArmy were airlifting VVIP and tourists, they left the civilians stranded and #Kashmiris helped their own. This is a fact!
#IndianArmy in its fresh act of state terrorism killed a number of civilians. #India refuse to resolve #kashmir dispute, @

Fig. 19 Snapshot of result of example query 4

order within database are designated as a Temporal Database. ZILD is a special kind of temporal database that stores data values with their associating timestamps to indicate the time interval of their validity, and also maintain the history of data objects based on their existence. It is competent to store various versions of the data based on time, which allow users to audit complete history of data object.

2. ZILD architecture contains mainly three components viz., "Temporal Database", "Query Store", and "Update Store". Temporal database contains time-stamped data, query store contains information about all queries executed, and update store contains information about all data update operations performed on the database. The proposed time-aware provenance framework has been developed around the concept of a Zero Information Loss Database (ZILD). Designing temporal relations by associating transaction or validation time for time stamping with all data values helps in evolution of historical database which maintains history of any data values whenever it is captured in the database with each transaction. In this way, ZILGDB provides data version support to maintain the history of all updates in form of provenance data along with the provenance of

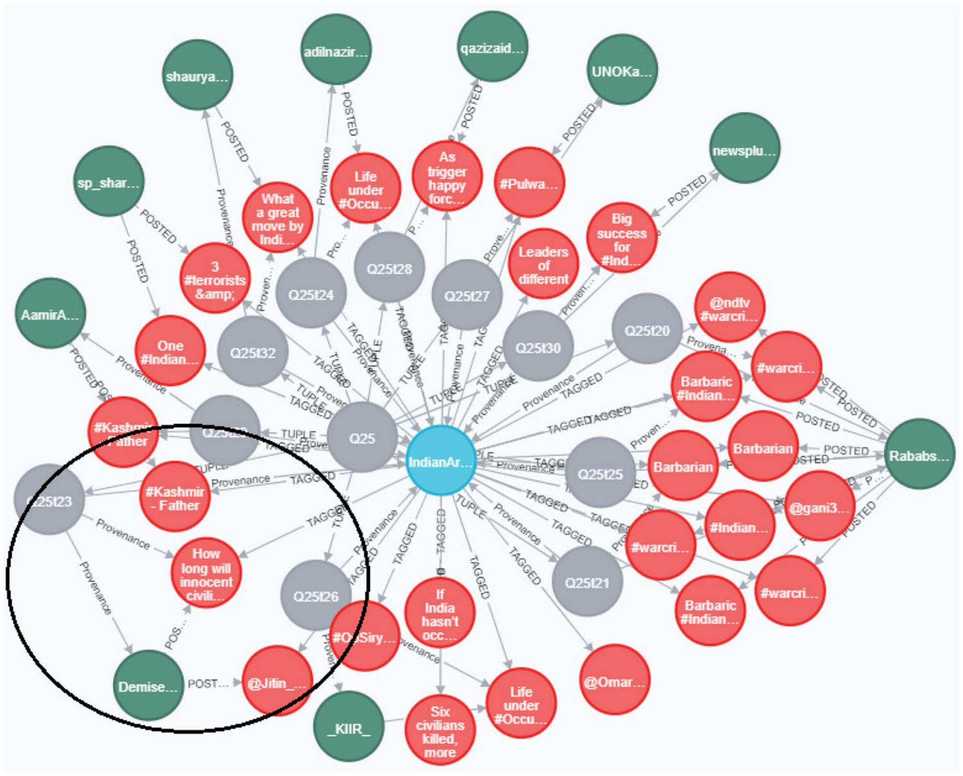


Fig. 20 Partial provenance graph of example query 4

all insert and delete operations, whereas no any existing provenance system supports update management.

3. Some of existing provenance system such as SFM (Kerchner et al. 2016), web-based provenance system (Gundecha et al. 2013b; Ranganath et al. 2013) captures only few pre-defined provenance attributes such as name, gender, religion, location etc., from different social accounts associated with a particular twitter user, but they neither provides a provenance path nor a propagation history and updates of published data like ZILGDB.
4. Few other provenance systems for example RAMP (Park et al. 2011), Hadoop-Prov (Akoush et al. 2013), PROV-SAID (Taxidou et al. 2015, 2018; De Nies et al. 2015) primarily focuses on workflow provenance. These systems are process-oriented that captures only coarse-grained information such as information about process and entities involved in that process whereas, ZILGDB captures detailed fine-grained provenance i.e. how any result is derived, what queries are executed, what operations are performed on data etc.
5. Mere capturing provenance information is of no use, until it is queried and/or visualized efficiently. Our proposed framework effectively supports multi-depth provenance querying with varying depth to know about direct or indirect sources contributed to a specific result. It captures both direct sources (i.e. Single-level provenance) and indirect sources (Multi-level provenance) of data that are contributing towards its generation along with the transformations applied on data, and the history of data.

6 Applications

Proposed framework is beneficial in attempting to understand the social processes and behaviour of a social media user. Some of the application scenarios are given below:

6.1 Use case: terrorist attack investigation

In this section, applicability of proposed SDP framework is presented for investigating terrorist attack, and identifying suspects, and their associated communities on social media especially in Twitter's network. A cyclic process model based on proposed framework is shown in Fig. 18. Initially, a large volume of twitter data set is fed into Neo4j graph database for performing further analysis based on suspicious tweets and hashtags. Whenever a user issues a query on database with specific predicates, a set of result tuples, based on issued query are generated. By employing the proposed provenance framework, a set of provenance graphs for all the result tuples are also generated, which consists of a set of nodes of the source graph, that are contributing to produce it. Now, these provenance graphs are required to be visualized and analyzed to obtain *Who*, *Why*, and *How* Provenance

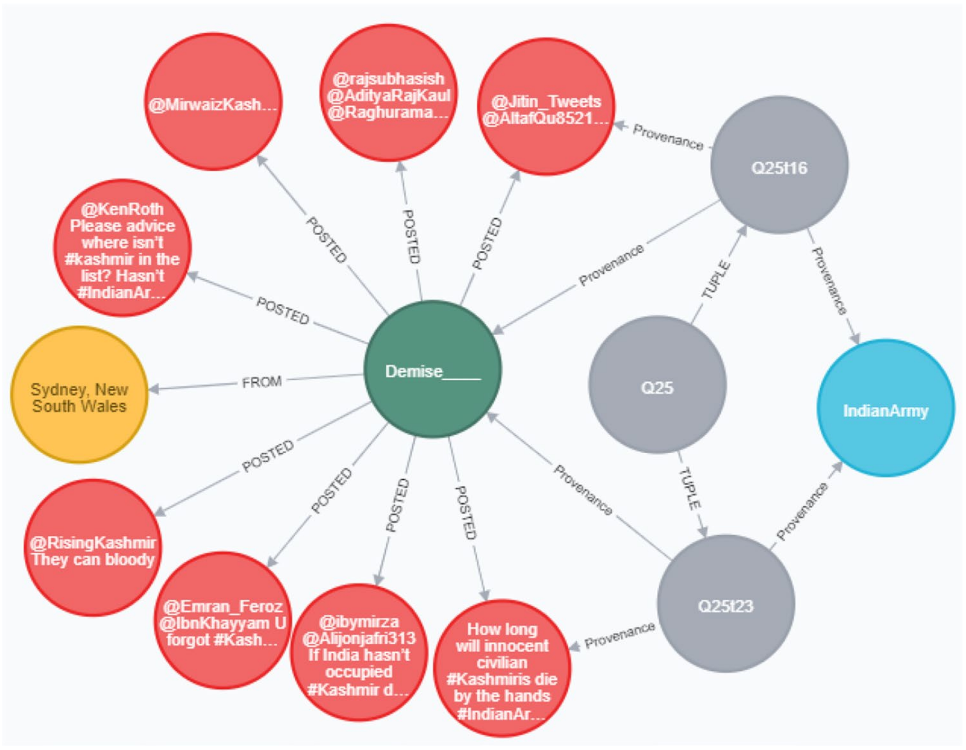

@MirwaizKash...
@rajsubhasish @AdityaRajKaul @Raghurama...
@Jitin_Tweets @AltafQu8521...
Provenance
Q25t16
@KenRoth Please advice where isn't #kashmir in the list? Hasn't #IndianAr...
POSTED
Provenance
TUPLE
Provenance
Sydney, New South Wales
FROM
Demise____
Q25
IndianArmy
@RisingKashmir They can bloody
@Emran_Feroz @IbnKhayyam U forgot #Kash...
@ibymirza @Alijonjafri313 If India hasn't occupied #Kashmir d...
How long will innocent civilian #Kashmiris die by the hands #IndianAr...
Q25t23

Fig. 21 Partial provenance graph

of a suspicious tweet exists in the result set. Further, all the other tweets posted by a suspicious user can be retrieved from the source database using proposed framework. This provenance data can be visualized in a way that may provide support back to the on-going investigation process. In this way, a user can visualize the provenance graph to identify such suspicious person and his linked communities.

A use case scenario is given below to explain the use of above process model in terrorist attack investigation by identifying suspicious person and his linked communities in Twitter's Network. A publicly available twitter data set related to an incidence of 11 killings at Pulwama district of Jammu and Kashmir on 15/12/2018 are used for the analysis purpose. The data set consists of around 10,000 tweets those are reactions to that incidence. Using cypher scripts, data set is first modelled into Neo4j graph database as per the data model explained in Sect. 4.1. Now, by employing the proposed provenance framework, our goal is to identify the suspects those are not in favour of Indian Army. Therefore, we have executed the following query to retrieve all the tweets that are posted on 15/12/2018 with a hashtag "IndianArmy".

Example query 4: Find all the tweets posted on 15/12/2018 in which hashtag "IndianArmy" is used.

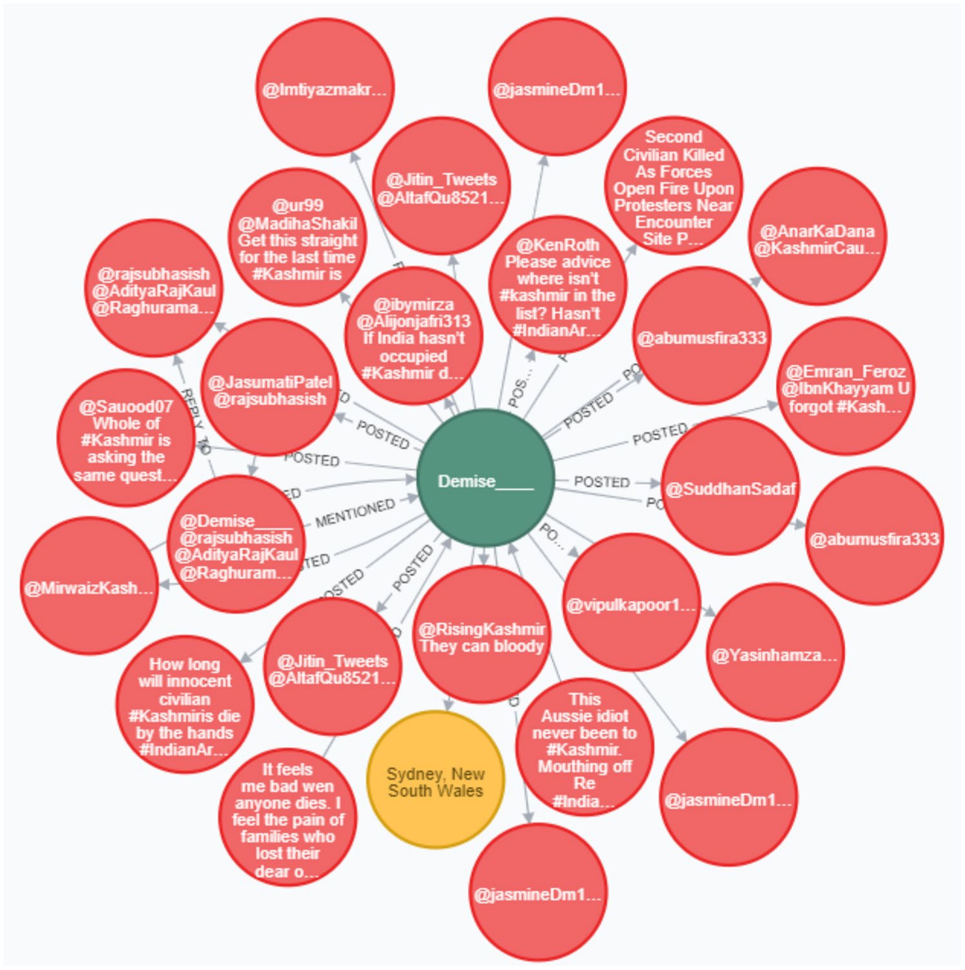
Cypher query 4: MATCH (y.YEAR)-[:Has_Month]->(m:MONTH)-[:Has_Day]->(d:DAY)-[:HAS_ TWEET]->(t:TWEET)<-[tg:TAGGED]-(h:HASHTAG) where h.Hashtagname = 'IndianArmy' and t.year=2018 and t.month=12 and t.day=15 return t.Tweet_Text as TWEET limit 100

In response to the above query execution, 53 result tuples and their provenance graphs are generated. After analyzing all the result tuples, suspicious or provoked tweets exist in the result set are marked. A snapshot of partial result set of this query execution is shown in Fig. 19, in which a highlighted tweet is looking suspicious or somewhat against the Indian Army.

Now, a provenance graph of this suspicious tweet as highlighted in Fig. 19 is required to visualize and analyze to obtain *Who*, *Why*, and *How* Provenance such as *Who* has posted this tweet?, *Why* this is present in the result set?, and *How* it is derived? etc. A partial provenance graph of above example query 4 is shown in Fig. 20, where provenance information of all the result tuples, i.e., grey nodes, are shown with their respective provenance edges, those are associated with a set of nodes of the source graph which contributed to produce it. The provenance graph of this identified suspicious tweet with result tuple id Q25t26 is mentioned with a marked portion in Fig. 20. After visualizing this provenance graph, we determined that this suspicious tweet is posted by a twitter user whose screen name is 'Demise_ _ _ _' as shown in Fig. 21. Now, on the basis of this screen name, we retrieved all the other tweets posted by this user in the source graph database as shown in Fig. 22.

Again, analyzing all the other tweets posted by this identified user during Pulwama incident held on 15/12/2018, we found that tweet's texts are looking suspicious or somewhat

Fig. 22 All Tweets posted by identified user 'Demise_ _ _ _'


Demise____
@Imtiyazmakr...
@jasmineDm1...
@Jitin_Tweets @AltafQu8521...
Second Civilian Killed As Forces Open Fire Upon Protesters Near Encounter Site P...
@ur99 @MadihaShakil Get this straight for the last time #Kashmir is
@KenRoth Please advice where isn't #kashmir in the list? Hasn't #IndianAr...
@AnarKaDana @KashmirCau...
@rajsubhasish @AdityaRajKaul @Raghurama...
@ibymirza @Alijonjafri313 If India hasn't occupied #Kashmir d...
@abumusfira333
@JasumatiPatel @rajsubhasish
@Emran_Feroz @IbnKhayyam U forgot #Kash...
@Sauood07 Whole of #Kashmir is asking the same quest...
@SuddhanSadaf
@abumusfira333
@Demise____ @rajsubhasish @AdityaRajKaul @Raghuram...
@MirwaizKash...
@vipulkapoor1...
@RisingKashmir They can bloody
@Yasinhamza...
How long will innocent civilian #Kashmiris die by the hands #IndianAr...
@Jitin_Tweets @AltafQu8521...
This Aussie idiot never been to #Kashmir. Mouthing off Re #India...
It feels me bad wen anyone dies. I feel the pain of families who lost their dear o...
Sydney, New South Wales
@jasmineDm1...
@jasmineDm1...
POSTED
MENTIONED
REPLY_TO

against the Indian Army and our country, which may provoke others also. Therefore, there is a need to retrieve all the tweets posted by this suspicious user on twitter's timeline along with his attributes such as Profile Creation Date, Nationality, Location, Sex, Friend's List, Follower's List etc., and his linked communities on Twitter and other social networking sites, for further analysis.

The schematic diagram of a cyclic process model based on proposed provenance framework to identify suspicious persons and their linked communities are shown in Fig. 18. The proposed framework works in following four steps for the above cyclic process model:

1. Modelling of Social Data: A large volume of social media data is fed into Neo4j graph database and modelled according to the proposed data model to generate provenance graphs for all the result tuples of a result set.
2. Analysis of Tweets: Determining suspicious or provoked tweets (As per the criteria of security agencies) exist in the result set.
3. Provenance Visualization: These provenance graphs consist of a set of nodes of the source graph, that are contributing to produce it. Moreover, these provenance graphs are further required to visualize to knowing about the Who, Why, and How Provenance of various suspicious tweets exists in the result set.
4. Identification of Suspicious Persons: Identification of suspicious person who posted those provoked tweets and to retrieve all the attributes of that suspicious person and his/her linked communities.

Currently, out of these four steps following three steps i.e. Modelling of Social Data, Provenance Visualization and Identification of Suspicious Persons are fully automated and performed by employing the proposed provenance framework. The only step i.e. Analysis of Tweets is performed manually by the security person as per their criteria to determine suspicious or provoked tweet. In future, we shall extend our provenance framework by applying Traditional Machine Learning models such as Support Vector Machines (SVM), Recurrent Neural Network (RNN), Convolutional

Neural Network (CNN) etc., to predict whether a tweet falls into the positive or negative sentiment.

In this way, framework has the capability to model a large twitter dataset into a Neo4j graph database to perform provenance visualization based on suspicious tweets and hashtags, which can be helpful for security agencies in investigating suspicious persons and their linked communities. Also, it will be helpful in attempting to understand the attitude and behaviour of social media users. The framework supports following key features viz., big data modelling, social media analytics, exploration and visualization etc. It can also be used to extract intelligence from social networking sites.

6.2 Other applications

- **Preserving Progressive User Profiles:** A social media user can update his/her profile or may add new information or remove any existing information at any time. Proposed framework is applicable for such applications to maintains all the data updates performed without losing any information.
- **In Covid-19 Pandemic:** In current pandemic situation of COVID-19, where health related data (such as Covid positives, Recovered, Vaccinated, Post Covid Symptoms etc.) is provided by almost all the countries across the world. Although, this data is valuable and scattered at different portals yet, it is necessary to identify the various sources and derivation history of such data regardless to knowing of their geographic location. In this situation, our proposed framework can be applied to capture/store/analyse provenance information for such data for a better understanding of current situation and in fighting against the COVID-19 pandemic.

7 Conclusions and future work

In this paper, we designed and implemented a Zero-Information Loss Graph Database (ZILGDB) on top of which a Social Data Provenance (SDP) Framework has been developed to capture and querying provenance for Twitter data set. The proposed framework is capable to capture fine-grained provenance for various query sets including select, aggregate, and data update queries with insert, delete, and update operations. It supports historical data queries, and querying through time using updates management in ZILGDB. It also provides support to a detailed provenance analysis through visualization along with efficient multi-depth querying to determine both direct and indirect sources of an information.

The proposed framework is efficient in terms of average execution time for capturing and storing provenance for select, and data update queries. However, a small execution overhead is measured for some aggregate queries, where the aggregation is performed on a larger number of input tuples. Proposed framework supports efficient provenance querying for both justifying answers of a query result, and historical data queries at an accepted level of precision. The proposed data model for social data in graph database is proven to be very efficient for range queries using timeline approach. Our proposed framework and provenance algorithm prove to be very promising in dealing with increasingly challenging issue of trust in social media. We conducted a real-life use case study to evaluate the usefulness of our framework in terrorist attack investigation, to identify suspicious persons and their linked communities on social media, particularly in Twitter's network.

However, currently SDP framework is implemented for a single node Neo4j Graph Database rather than for several distributed nodes in a cluster.

In future, we plan to further extend our social data provenance framework for distributed graph database on different nodes in a cluster using Neo4j Aura.

References

Afra S, Alhajj R (2021) Integrated framework for criminal network extraction from Web. J Inf Sci 47(2):206–226

Akoush S, Sohan R, Hopper A (2013). Hadoopprov: towards provenance as a first class citizen in mapreduce. In: Proceedings of 5th USENIX workshop on the theory and practice of provenance (TaPP 13)

Allen D, Hodler A, Hunger M, Knobloch M, Lyon W, Needham M, Voigt H (2019) Understanding trolls with efficient analytics of large graphs in neo4j. In: Proceedings of Datenbanksystem for business, technologies and web (BTW 2019)

Angles R, Gutierrez C (2008) Survey of graph database models. J ACM Comput Surv (CSUR) 40(1):1–39

Angles R, Gutierrez C (2018) An introduction to graph data management. In: Graph data management. Springer, Cham, pp 1–32

Aryono T (2016) Modelling social media semi-structured data with graph database. In: Proceedings of international conference ICONIET, pp 1–7. https://www.academia.edu/27198471/Modelling_Social_Media_Semi_structured_Data_with_Graph_Database

Baeth MJ, Aktas MS (2017) A large scale synthetic social provenance database. In: Proceedings of the 9th international conference DBKDA, pp 16–22

Bearman DA, Lytle RH (1985) The power of the principle of provenance. Archivaria 1(21). http://journals.sfu.ca/archivar/index.php/archivaria/article/viewArticle/11231

Bhargava G, Gadia SK (1993) Relational database systems with zero information loss. J IEEE Trans Knowl Data Eng 5(1):76–87

Boselli R, Cesarini M, Mercorio F, Mezzanzanica M, Vaccarino A (2017, July) A pipeline for multimedia Twitter analysis through graph databases: preliminary results. In: Proceedings of international conference DATA, pp 343–349

Buneman P, Davidson SB (2010, September) Data provenance—the foundation of data quality. In: Proceedings of workshop: issues and opportunities for improving the quality and use of data within the DoD, Arlington, USA, pp 26–28

Buneman P, Tan WC (2019) Data provenance: what next? ACM SIGMOD Rec 47(3):5–16

Buneman P, Khanna S, Tan WC (2000, December) Data provenance: some basic issues. In: Proceedings of international conference on foundations of software technology and theoretical computer science, pp 87–93

Cattuto C, Quaggiotto M, Panisson A, Averbuch A (2013, June) Time-varying social networks in a graph database: a Neo4j use case. In: Proceedings of first international workshop on graph data management experiences and systems, pp 1–6

Cheney J, Chong S, Foster N, Seltzer M, Vansummeren S (2009, October) Provenance: a future history. In: Proceedings of the 24th ACM SIGPLAN conference companion on object oriented programming systems languages and applications, pp 957–964

Cheng Y, Nguyen D, Bijon K, Krishnan R, Park J, Sandhu R (2012, September) Towards provenance and risk-awareness in social computing. In: Proceedings of the first international workshop on secure and resilient architectures and systems, pp 25–30

Corsar D, Markovic M, Edwards P (2016, June) Social media data in research: provenance challenges. In: Proceedings of international provenance and annotation workshop (IPAW), pp 195–198

De Nies T, Taxidou I, Dimou A, Verborgh R, Fischer PM, Mannens E, Van de Walle R (2015, October) Towards multi-level provenance reconstruction of information diffusion on social media. In: Proceedings of the 24th ACM international on conference on information and knowledge management, pp 1823–1826

DeBoer D, Zhou W, Singh L (2013, June) Using substructure mining to identify misbehavior in network provenance graphs. In: Proceedings of the first international workshop on graph data management experiences and systems, pp 1–6

Duong CT, Nguyen QVH, Wang S, Stantic B (2017, September) Provenance-based rumor detection. In: Proceedings of Australasian database conference, pp 125–137

Durand GC, Pinnecke M, Broneske D, Saake G (2017, March) Backlogs and interval timestamps: building blocks for supporting temporal queries in graph databases. In: Proceedings of EDBT/ICDT workshops

Feng Z, Gundecha P, Liu H (2018) Social provenance. Springer, New York, pp 2768–2772

Fernandes D, Bernardino J (2018, July) Graph databases comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB. In: Proceedings of international conference DATA, pp 373–380

Filgueira R, Krause A, Atkinson M, Klampanos I, Spinuso A, Sanchez-Exposito S (2015, August) dispel4py: an agile framework for data-intensive escience. In: Proceedings of IEEE 11th international conference on e-Science, pp 454–464

Glavic B, Miller RJ (2011) Reexamining some holy grails of data provenance. In: TaPP 11

Gundecha P, Feng Z, Liu H (2013a, October) Seeking provenance of information using social media. In: Proceedings of the 22nd ACM international conference on information and knowledge management, pp 1691–1696

Gundecha P, Ranganath S, Feng Z, Liu H (2013b, August) A tool for collecting provenance data in social media. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining, pp 1462–1465

Herschel M, Diestelkämper R, Lahmar HB (2017) A survey on provenance: what for? What form? What from? VLDB J 26(6):881–906

Kaplan AM, Haenlein M (2010) Users of the world, unite! The challenges and opportunities of Social Media. J Bus Horiz 53(1):59–68

Kerchner D, Littman J, Peterson C, Smallen V, Trent R, Wrubel L (2016) The provenance of a tweet. https://scholarspace.library.gwu.edu/downloads/h128nd689

Markovic M, Edwards P, Corsar D (2013) A role for provenance in social computation. In: Proceedings of the first international workshop on crowdsourcing the semantic web—CrowdSem

Namaki MH, Song Q, Wu Y, Yang S (2019) Answering Why-questions by exemplars in attributed graphs. In: Proceedings of the international conference on management of data (SIGMOD '19)

O'Reilly T, Milstein S (2011) The Twitter book. O'Reilly Media, Inc., Newton

Papavasileiou V, Yocum K, Deutsch A (2019, June) Ariadne: online provenance for big graph analytics. In: Proceedings of the international conference on management of data, pp 521–536

Park H, Ikeda R, Widom J (2011) Ramp: a system for capturing and tracing provenance in mapreduce workflows. Proc VLDB Endow 4(12):1351–1354

Ramusat Y, Maniu S, Senellart P (2018) Semiring provenance over graph databases. In: Proceedings of 10th USENIX workshop on the theory and practice of provenance (TaPP 18)

Ranganath S, Gundecha P, Liu H (2013, October) A tool for assisting provenance search in social media. In: Proceedings of the 22nd ACM international conference on information and knowledge management, pp 2517–2520

Rani A, Goyal N, Gadia SK (2015, October) Data provenance for historical queries in relational database. In: Proceedings of the 8th annual ACM India conference, pp 117–122

Rani A, Goyal N, Gadia SK (2016, October) Efficient multi-depth querying on provenance of relational queries using graph database. In: Proceedings of the 9th annual ACM India conference, pp 11–20

Rani A, Goyal N, Gadia KS (2021) Provenance framework for Twitter data using zero-information loss graph database. In: Proceedings of the 8th ACM IKDD CODS and 26th COMAD, pp 74–82

Riveni M, Baeth MJ, Aktas MS, Dustdar S (2017, August) Provenance in social computing: a case study. In: Proceedings of the 13th international conference on semantics, knowledge and grids (SKG), pp 77–84

Robinson I, Webber J, Eifrem E (2015) Graph databases: new opportunities for connected data. O'Reilly Media, Inc., Newton

Sharma S (2015) An extended classification and comparison of nosql big data models. arXiv preprint arXiv:1509.08035

Silberschatz A, Korth HF, Sudarshan S (1996) Data models. J ACM Comput Surv (CSUR) 28(1):105–108

Simmhan YL, Plale B, Gannon D (2005) A survey of data provenance in e-science. Proc ACM Sigmod Rec 34(3):31–36

Soni D, Ghanem T, Gomaa B, Schommer J (2019, June) Leveraging Twitter and Neo4j to Study the Public Use of Opioids in the USA. In: Proceedings of the 2nd joint international workshop on graph data management experiences & systems (GRADES) and network data analytics (NDA), pp 1–5

Soto A, Ryan C, Peña Silva F, Das T, Wolkowicz J, Milios E, Brooks S (2018) Data quality challenges in Twitter content analysis for informing policy making in health care. In: Proceedings of Hawaii international conference on system sciences (HICSS)

Tas Y, Baeth MJ, Aktas MS (2016, August) An approach to standalone provenance systems for big social provenance data. In: Proceedings of the 12th international conference on semantics, knowledge and grids (SKG), pp 9–16

Taxidou I, De Nies T, Verborgh R, Fischer PM, Mannens E, Van de Walle R (2015, May) Modeling information diffusion in social media as provenance with W3C PROV. In: Proceedings of the 24th international conference on world wide web, pp 819–824

Taxidou I, Lieber S, Fischer PM, De Nies T, Verborgh R (2018) Web-scale provenance reconstruction of implicit information diffusion on social media. J Distrib Parallel Databases 36(1):47–79

Twitter Data Set (2018) https://www.kaggle.com/umarhabib/pulwama-killing-twitter-data

Wang J, Crawl D, Purawat S, Nguyen M, Altintas I (2015, October) Big data provenance: challenges, state of the art and opportunities. In: Proceedings of the IEEE international conference on big data (big data), pp 2509–2516

Yang J, Yu M, Qin H, Lu M, Yang C (2019) A Twitter data credibility framework—Hurricane Harvey as a use case. ISPRS Int J Geo-Inf 8(3):111

Yuan Z, Ton That DH, Kothari S, Fils G, Malik T (2018) Utilizing provenance in reusable research objects. J Inform 5(1):14

Zhang E, Fiaidhi J, Mohammed S, Rd O, Bay T, Pb ON (2017) Social recommendation using graph database Neo4j: mini blog, Twitter social network graph case study. Int J Future Gener Commun Netw 10(2):9–20

Zhao L, Hua T, Lu CT, Chen R (2016) A topic-focused trust model for Twitter. J Comput Commun 76:1–11